LANCASTER UNIVERSITY

School of Computing and Communications

# A Transfer Learning-aided Decision Support System for Multi-Cloud Brokers

by

Faiza Samreen

MS in Computer Science, International Islamic University, 2007

Bachelor of Computer Science (Hons), International Islamic University, 2003

**Thesis submitted for the degree of**

***Doctor of Philosophy***

**June 2017**

ABSTRACT

## A Transfer Learning-aided Decision Support System for Multi-Cloud Brokers

*by Faiza Samreen*

MS in Computer Science, International Islamic University, 2007

Bachelor of Computer Science (Hons), International Islamic University, 2003

Thesis submitted for the degree of Doctor of Philosophy

June 2017

School of Computing and Communications

Lancaster University

Decision-making in a cloud environment is a formidable task due to the proliferation of service offerings, pricing models, and technology standards. A customer entering the diverse cloud market is likely to be overwhelmed with a host of difficult choices in terms of service selection. This applies to all levels of service, but Infrastructure as a Service (IaaS) level is particularly important for the end user given the fact that IaaS provides more choices and control for application developers. In the IaaS domain, however, there is no straightforward method to compare virtual machine performance and, more generally cost/performance trade-offs, within or across cloud providers. A wrong decision can result in a financial loss as well as a reduced application performance. A cloud broker can help in resolving such issues by acting as an intermediary between the cloud provider and the cloud consumer – hence, serving as a decision support system for assisting the customer in the decision process.

In this thesis, we exploit machine learning for building an intelligent decision support system which assists customers in making application-driven decisions in a multi-cloud environment. The thesis examines a representative set of appropriate inference and prediction based learning techniques, that are essential for capturing application behaviour on different deployment setups, such as Polynomial Regression and Support Vector Regression (SVR). In addition, the thesis examines the efficiency of the learning techniques, recognising that machine learning can impose significant training overhead. The thesis also introduces a novel transfer learning aided technique, leading to substantial reduction in this overhead. By definition, transfer learning aims to solve the new problem faster or with a better solution by using the previously learned knowledge. Quantitatively, we observed a reduction of approximately 60% in the learning time and cost by transferring the existing knowledge about the application and cloud platform in order to learn a new prediction model for some other application or cloud provider. Intensive experimentation has been performed in this study for learning and evaluation of proposed decision support system. Explicitly, we have used three different representative applications over two cloud providers, namely Amazon and Google. Our proposed decision support system, enriched with transfer learning methods, is capable of generating decisions that are viable across different applications in a multi-cloud environment. Finally, we also discuss lessons learned in terms of architectural principles and techniques for intelligent multi-cloud brokerage.

# Declaration

I declare that the work in this thesis is my own work and has not been submitted either in whole or in part for the award of a higher degree elsewhere. Any sections of the thesis which have been published are clearly identified.

................

Faiza Samreen

## Acknowledgements

When I first joined the Ph.D. program at the Lancaster University back in 2013. I could not fathom what it would feel like to successfully defend my thesis so many years later. I have been so lucky to be accepted into such a prestigious university as LU. It was such an adventure moving from Pakistan to the beautiful city of Lancaster, and I know that no matter where I am my heart will always have a special place for the Windy City. So it is with a sense of bittersweet accomplishment that I sit today to type up my Acknowledgements and add the final touches to my thesis just hours before my final submission.

I think all Ph.D. graduate students go through the same highs and lows when they are working on their research. But having a good supervisor is probably the only thing that can actually make you come out stronger when you feel you have hit a wall and you don't know which direction to turn.

I have been extremely fortunate with my supervisors, Professor. Gordon Blair and Dr. Yehia Elkhatib. Professor. Gordon Blair is one of the nicest professors I know. Every time I would leave his office after a lengthy meeting I would think to myself, "I hope I can be as good a supervisor as he is one day." His thoroughness and attention to detail always inspired me. From our talks and discussions, I learned many things other than about my research that I will remember for years to come. With him, I never felt that I was "just another student". He is extremely kind and considerate, and he does what it takes to make his students come out shining on top no matter how far behind they were when they started. I would say to all students who are lucky enough to have him as their supervisor that they could not be in better hands.

I would like to thank Dr. Yehia Elkhatib, who was always there to keep me on my toes. If not for him keeping me on track I would probably be stuck in a dead-end idea. Nothing I write here can come close to expressing the gratitude I have for the time and effort he expended on me the last 4 years. He is professional, yet he is also very understanding and he knew how to keep me motivated by telling me exactly what I needed to hear to stay focused and to not lose sight of what my aims were.

I acknowledge partial funding by COMSATS Institute of Information Technology, Amazon Web Services educational research grant, Graduate School travel grant, PEEL studentship and William Ritchie & Friends Program for the financial help to assist my study.

I am really very grateful to Dr. Vatsala Nandllol, Irni Khairuddin, Roberto Rodrigues and Izhar Ullah for always standing by my side whenever I had needed them.

I thank my dear friend Zunaira, who was always there to answer my frantic emails or phone calls about the challenges I was running into with my research. In her usual calm way, she would always give me good advice on how to pull through the difficulties and luckily things always worked out better than I expected.

I would like to thank Fatimah, Naheed, and Sana, my best friends with whom I probably shared the best times of my life before coming to LU. They were always there to make me laugh through whatever crisis I felt I was going through, be it personal or professional. I am very grateful to Aleena, Aneesa, Sadia and Maheen for their continuous support.

I would like to thank my dear brother Dr. Asif, and my dear sisters Dr. Humera, Dr. Sumera and Dr. Sadia, who always motivated and supported me during my time at LU. They have always been there to make me cheerful by sharing the loving memories of our childhood. They are very close to my heart and my life is incomplete without them. I thank Amna, Ayaan, Hamnah, Talha and Taaha for adding beautiful memories in our lives.

I would like to thank my dearest husband, Ahmad Waseem, who was always there to help me and to make me happy. He is the best partner anyone can wish for. His calmness, kindness, and tolerance helped me to finish my thesis in great spirits. Along with the many tiring hours, I spent in front of my computer typing up my thesis, I had many fun hours in front of T.V watching countless movies and Hasb-e-Hall with him. He has contributed as much to this thesis as I have just by being by my side. He is probably the only one who could truly understand my feelings of despair. Knowing that we both were in the same boat really helped get us through the tough times together!

My special gratitude for my parents, who have always been my great support. Their prayers, encouragement and spectacular advice were always with me. I can never forget what they have taught me nor can I stop loving them and thanking them for all that they have done for me. I am lucky enough to have parents like them who have not let a single second pass without loving me unconditionally. May Allah bless them always.

Lastly, I would like to say thanks to my lovely son Umar, who filled my life with happiness. My dear son Umar! when I tell you I Love You, I say it to remind you that *you* are the *Best Thing* that ever happened to me.

Faiza Samreen

*To my parents, Dr. Muhammad Alim and Fayyaz Akhtar.*

Without the inspiration, determination, and support that you have given me,

I might not be the person I am today.

# Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

## 1.1 Overview

The cloud computing market is growing incredibly fast. In 2013, the global application market had a value of 30.35 billion U.S. dollars which kept on increasing every year and had reached to 62.5 billion U.S. dollar in 2017 [1]. According to Gartner's lastest worldwide public cloud services revenue forecast, the market is projected to grow 18.5% in 2017 reaching $260.2 billion, up from $219.6 bilion in 2016 [2]. Consequently, decision-making in a cloud environment is a challenging task due to the proliferation of service offerings, pricing models, and technology standards [3]. This applies to all levels of service, but the Infrastructure as a Service (IaaS) level is particularly important for the end user given the fact that IaaS provides more choices and control for application developers. In the IaaS domain, there is no straightforward method to compare virtual machine performance and, more generally cost/performance trade-offs, within or across cloud providers. A wrong decision can result in financial loss as well as reduced application performance [4, 5, 6, 7]. A cloud broker can help in resolving such issues by acting as an intermediary between the cloud provider and the cloud customer – hence, serving as a decision support system for assisting the customer in the decision process [8].

The thesis investigates the problem of decision making to assist cloud customers in finding an optimal IaaS deployment strategy based on application-specific requirements and customer-related QoS constraints. The thesis argues for the use of machine learning methods for developing an intelligent decision support system. We believe that machine learning can provide behavioural and performance insights about the application and deployment setup necessary to make optimal decisions in terms of performance and cost. Recognising that machine learning can impose significant training overhead, a key contribution of this thesis is a transfer learning aided methodology for efficient decision making, thus making this approach more cost-effective for cloud brokers.

The remainder of this chapter is organised as follows. The chapter starts with a discussion about the significance of decision support systems. Following that, the chapter outlines the key underline concepts of cloud brokers, decision support systems and machine learning and explains their relationship in the context of cloud computing. Moreover, the chapter discusses the limitations of the current state of the art. Further to that, the chapter presents the research goals of the thesis, research methodology and contributions. Lastly, the chapter outlines the structure of the thesis.

## 1.2 Motivation

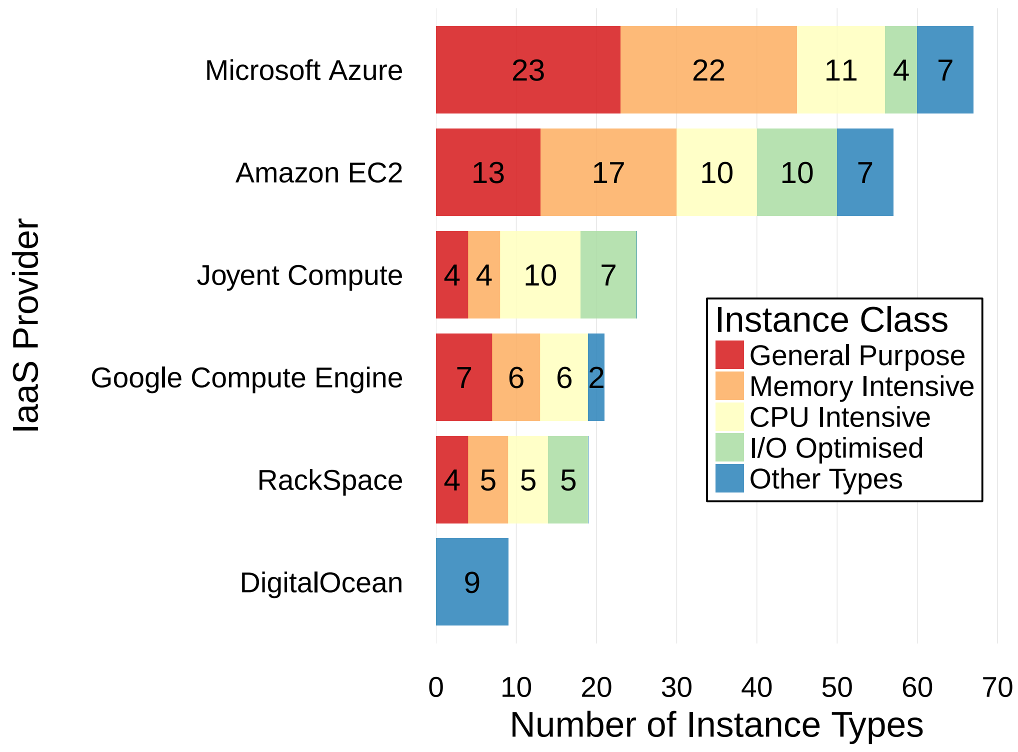


**Figure 1.1:** The number of Linux-based instance types offered by major IaaS providers, as of July 2017.

Figure 1.1 depicts an overview of the cloud market in July 2017, showing the offered instance types under different categories by major IaaS providers. The various colours in the graph indicate the categories of Linux-based instance types (virtual machines) and the horizontal axis depicts the number of instance types in each category. This data is collected manually, based on market leading cloud service providers. It is evident that most IaaS vendors offer 20 or more instance types under different categories. Microsoft Azure is offering more than 60 different instance types, the maximum of all the listed IaaS providers.

IBM Softlayer is offering even more choices by providing an interface to the cloud customers where they can use a slider to build a virtual server that fits their needs, as shown in Figure 1.2. The customers can choose hourly or monthly billing. In addition, they can opt for additional services such as extra storage, firewall provision, monitoring, and various operating systems according to their choice. Such an interface offers hundreds of different types of virtual machine setup permutations. No one can imagine how challenging it would be to select from such a pool of offered instance types. There is a clear need for tools to simplify selection decision and potentially automate such decision making.

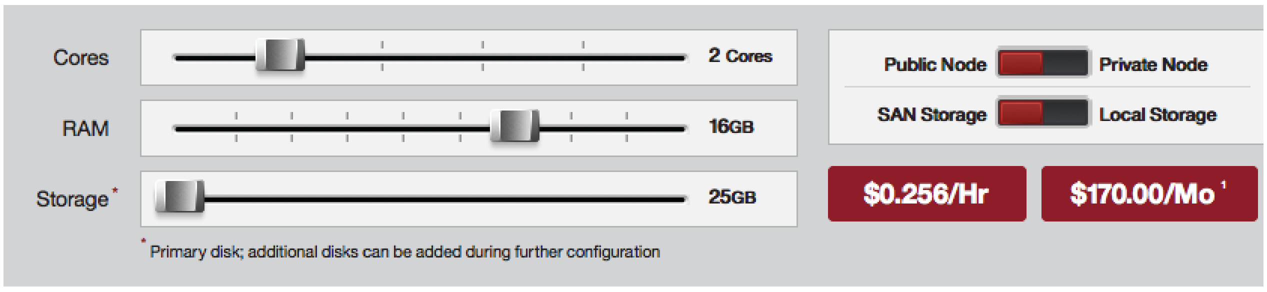

Cores 2 Cores
RAM 16GB
Storage* 25GB
* Primary disk; additional disks can be added during further configuration
Public Node / Private Node
SAN Storage / Local Storage
$0.256/Hr
$170.00/Mo [1]

**Figure 1.2:** IBM Softlayer configurations, as of 2017

## 1.3 Background

### 1.3.1 Cloud Computing

Cloud computing is not a new technology; rather it has evolved from various existing technologies such as grid computing, utility computing, virtualisation and autonomic computing [9]. Cloud computing is an exciting technological model where resources, such as CPU time, storage and bandwidth are provided as general utilities for lease by users over the internet in an on-demand fashion. Cloud computing has had a tremendous impact on the IT industry with its features of lowering operational cost, high scalability, zero upfront investment and reduction of maintenance expenses. Large companies such as Amazon, Google, Microsoft and others are striving to provide reliable and cost-efficient cloud platforms to users.

Cloud providers can be classified as public or private. Public cloud providers offer their resources as a service to the general public and charge them under a specified pricing model for utilised resources. Private clouds are mainly designed for the internal use of an organization to get full control of security, performance, and reliability.

Cloud providers offer their services at three main levels: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) [10]. IaaS refers to the provisioning of computer infrastructure (such as platform virtualisation), storage and networking. These offerings have heterogeneous configuration options (e.g. different choices of virtual machine configurations in terms of memory, CPU, I/O, etc), availability zones and so on. Examples of IaaS are Amazon EC2 and the Google Compute Engine. PaaS provides platform layer resources such as operating system support and software development frameworks. Google App Engine, Microsoft Windows Azure, and Force.com are some examples. SaaS refers to providing turn-key applications over the internet such as Salesforce.com.

Cloud providers use a variety of pricing models, including usage-based fixed pricing, usage-based dynamic pricing, subscription-based pricing, reserved services contracts with a combination of usage-based fixed pricing and up-front fees, auction-based pricing and so on [11]. In addition to that, different cloud providers use different pricing schemes. For instance, Amazon charges for virtual machines on an hourly basis, Microsoft Azure bills on a per minute basis while Google charges a minimum of 10 minutes per virtual machine.

### 1.3.2 Cloud Brokers

A cloud broker is an important supporting actor in the cloud environment, acting as an intermediary between the provider and the consumer [12]. According to NIST, a cloud broker is defined as, “an entity that manages the use, performance, and delivery of cloud services and negotiates relationships between the cloud providers and the cloud consumers” [13]. Cloud brokers participate in a cloud computing environment with various roles such as intermediation, aggregation, arbitrage and integration [14].

A large body of research has been carried out to provide brokerage solutions such as abstraction and interoperability to reduce deployment complexity. In addition, there has been research investigating the management of multi-cloud environment. Here, the term "multi-cloud" denotes the usage of multiple independent clouds by a client or service. Moreover, there has been a small amount of research looking at decision making to assist customers in the selection process by ranking cloud providers and their offered services.

Brokerage solutions can be classified as either hosted or deployable services. Hosted services are externally managed by a third party and customers have no insight as to how the application is provisioned. RightScale Cloud Portfolio Management [15], anStratus [16], xStream [17] and Cliqr [18] are some of the examples of a hosted service. In contrast, deployable services rely on open source software solutions that can be operated either internally by a corporation or externally as a grey-box service. Deployable approaches intend to be transparent for the application developer in order to feed the deployment and scalability policies. Examples include Apache Brooklyn [19], Scalr[20], Standing Cloud [21] and Aelous [22]. The mentioned solutions tackle interoperability and abstraction to reduce the deployment complexity of applications. However, except for RightScale, none of these solutions provide assistance for cloud selection.

### 1.3.3 Decision Support Systems

The cloud computing market is crowded with a large number of public IaaS providers [1, 2]. Therefore, a customer entering the cloud market has to face difficult comparisons and complex decisions. One way to address this challenge is to explore all the possible options exhaustively, which is quite an expensive and time-consuming task. Another way is to select a provider based on their reputation and then choose the resources from their suggested VM options. Such selections are not necessarily an effective choice for optimised performance or cost. The service offerings of cloud providers are somewhat of a black box for customers as they are not aware of the necessary details required for selection. Therefore, decision support is considered one of the fundamental objectives of cloud brokerage. However, current cloud broker frameworks tend to focus on abstraction, interoperability, and policy management. However, they are lacking support in automated decision-making and hence the selection of IaaS resources is mostly left to the customer.

A decision support system has to be responsive in order to satisfy customer-defined policies throughout the application life cycle. STRATOS [23], MODACloud [24], Cloud4SOA [25], Cloudcmp [26] and Broker@Cloud [27] have taken some preliminary steps in this regard. The decisions are typically based on the ranking of cloud providers or services taking into account the non-functional requirements and ignoring variability in application behaviour and cloud resource. Apart from ranking, some of the decisions are derived from a synthetic workload that cannot portray the real performance variation of the cloud [26, 28, 29]. Overall, decision making is challenging due to issues such as adaptability and also the need to satisfy multi-objective requirements from cloud customers [27].

Machine learning can help address such complicated decisions by observing application behaviour and modelling relationships amongst independent sources of knowledge such as cloud resource configuration and application behaviour. Therefore, machine learning has the potential to provide an intelligent and grounded decision support system. CloudProphet [28], Matrix [30] and [31] have taken first steps towards the use of machine learning in this context. These solutions are not considered a cost-effective option due to the requirement for a large amount of training data for the model generation. Furthermore, the learning models are not generalised enough to be used for different application types or Quality of Service (QoS) attributes. Despite such considerable efforts, a decision support system in a multi-cloud environment is still an under-researched and demanding area that needs to be explored.

## 1.4 Limitations of Existing Approaches

The research in this thesis sits at the intersection of the three areas studied above, namely cloud brokers, decision support systems and machine learning (see Figure 1.3). Area 'A', indicates the area of work where cloud brokers mainly support interoperability, abstraction and unification of multi-cloud systems. Area 'B' focuses on areas such as policy and SLA management regardless of cloud broker concepts. Area 'C' captures the research where machine learning is taking advantages of cloud computing to deal with big data analytic. Area 'F' represents decision making in a cloud environment where machine learning is used to provide solutions to many problems such as scheduling, load balancing and energy consumption. There is barely any research effort (area 'G') to exploit machine learning to assist the functionality of cloud brokers. A few solutions are available from the cloud broker community to assist the customer with deployment-related decisions and are indicated in area 'D'. Most notably, there is very little research at the intersection of all three areas (area 'E') which is the focal point of this research.

To date, there have been no successful integrated framework providing an end-to-end solution to

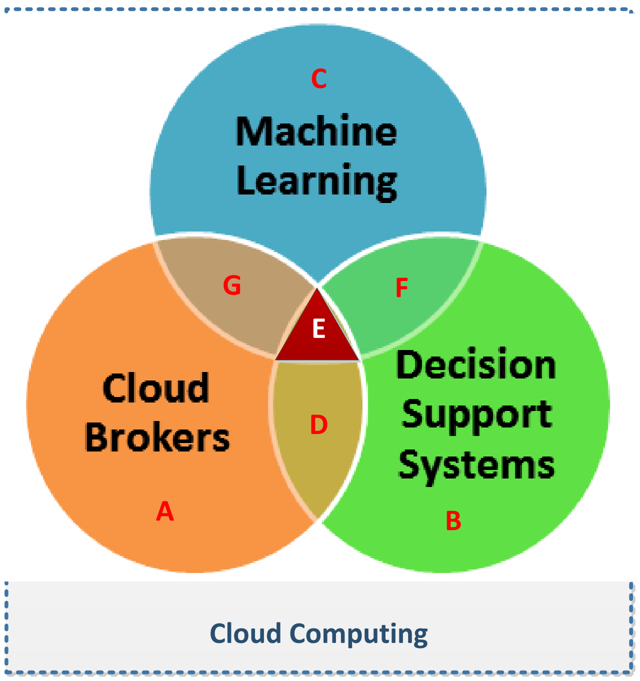


**Figure 1.3:** Area of study

assist customers with optimal deployment choice. Moreover, the available methods are designed to deal with a specific application or single cloud infrastructure. In addition, there is a lack of solutions that consider support for a range of application types across multiple cloud providers.

## 1.5 Statement of Research

This thesis suggests that intelligent decision support systems integrated with cloud brokerage are still an under-researched area. Likewise, there is still a long way to go in terms of providing efficient decision making to help the customer in terms of optimised and application-driven deployment decisions in a multi-cloud environment.

### 1.5.1 Goals & Research Questions

The key goal of the work presented in this thesis is to investigate the role of machine learning in developing a decision support system integrated with a cloud brokerage solution in a multi-cloud environment. More precisely, the end goal is a decision support system to assist customers in making deployment decisions by taking into account the application requirements and customer constraints.

This overall goal can be further decomposed into three inter-dependent research goals.

1. The designing of a cloud broker architecture integrated with an implementation of an intelligent decision support system.

   The key research questions associated with this goal are:

   (a) What is the potential role of an intelligent decision support system in enhancing cloud brokerage?

   (b) What does an intelligent decision support system architecture look like and which of its components is responsible for offering an understanding of application behaviour?

2. The investigation of machine learning methods that can be applied for optimal decision making in the decision support system of a cloud broker.

   The key research questions associated with this goal are:

   (a) What machine learning methods are most applicable to support decision making in the cloud?

   (b) How can machine learning methods provide the insight necessary to help decision support systems? And to what extent can generalised learning models be used for learning across different application domains?

3. The development and evaluation of an efficient decision-making method integrated with the established decision support system (assuming that the research goal 2 is viable) to reduce the learning and decision-making cost and to making it more cost-effective for use in cloud brokers.

   The key research questions associated with this goal are:

   (a) What are appropriate methods to make use of existing knowledge?

   (b) How can learning be done efficiently to overcome the challenge of additional time and learning cost?

### 1.5.2 Research Methodology

This work adopts an *experimental* systems research methodology, with an *iterative* approach that is based on *quantitative* analysis of real systems.

*Experimental:* In order to investigate the role and effectiveness of the machine learning for a decision support system, a prototype named Daleel is developed to iteratively test different learning models. A large-scale analysis is used to incrementally enhance the learning model in order to make it more effective in a generalised manner. This is shown in Figure 1.4, where the analysis results are to fed iteratively into the design of learning models. Daleel requires generalised learning model(s) to be used for decision making.

*Iterative:* A learning method is selected by exploring different traditional machine learning techniques. The exploration of different machine learning methods is used to build the learning models to be eventually used, keeping in mind the need for getting insight about the application behaviour with respect to the deployment setups. The intuition behind exploring different learning methods is the use of diverse baseline models that can be used for diagnostic and accuracy assessment as well. The learning models are built and refined following circular processes as shown in the middle green circle in Figure 1.4. The process of getting a final learning model starts with the selection of learning method followed by fitting a model. The fitted model is then evaluated on analysed data and further examined using statistical methods for update and re-fit. A human intervention is involved at this level in order to visually analyse the output of statistical results. By following this iterative process and further evaluating it with incremental data analysis a final model for decision making is derived. A detailed experimental study is done to check the feasibility of offering different learning models which are discussed in detail using different visualisations.

*Quantitative:* Furthermore, our research methodology follows a quantitative assessment of real world scenarios in a real-time environment. A quantitative analysis is performed on a large scale experimental setup of two major IaaS providers; the first one is EC2 of Amazon Web Services and second is Google Compute Engine (GCE). For evaluation, three real-time applications of different categories are used as a representative subset of problems, as depicted in the blue rectangle at the top of Figure 1.4. The real-time data traces of the applications, running on different deployment setups (EC2 and GCE), are used for training and assessment of the prediction models. Therefore, the knowledge-base is comprised of real-time workload traces instead of synthetic workloads.

### 1.5.3 Contributions

This thesis investigates the problem of decision making in a multi-cloud environment and examines how an intelligent decision support system integrated with cloud brokerage can benefit the cloud

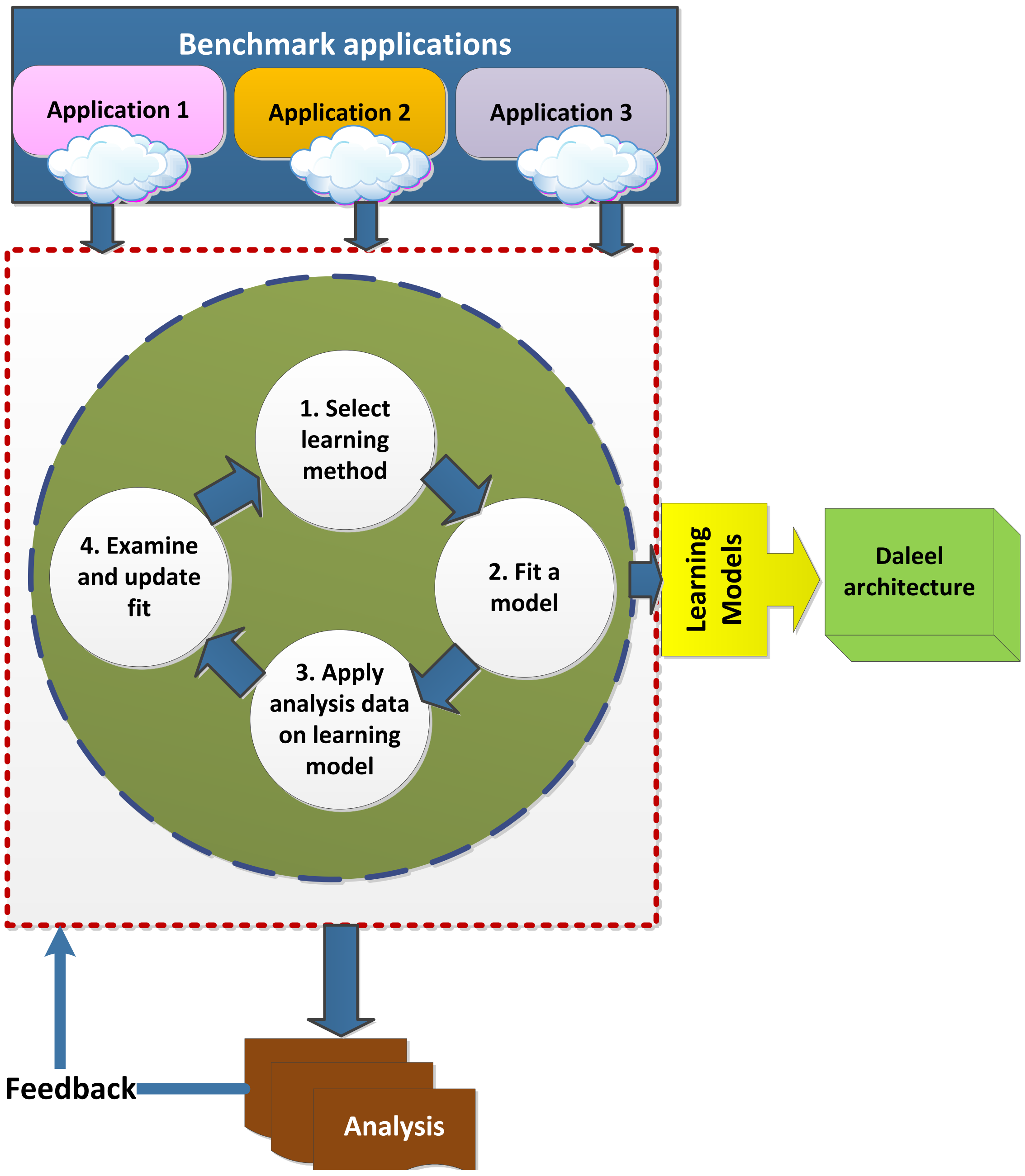


**Figure 1.4:** Research methodology approach

customer for making deployment decisions. The main contributions of this thesis are:

1. The architectural insight of a decision support system for inclusion of the machine learning methods and its integration with multi-cloud brokerage.

2. Model fitting engine equipped with generalised as well as application specific learning models.

3. A two-mode transfer learning scheme for efficient decision making. This scheme is based on transferring knowledge from one domain to the other. The efficiency is achieved by reducing the learning overhead in terms of time and cost. Quantitatively, an overall reduction of 60% is observed.

Additional contributions of this thesis are:

1. A detailed study of machine learning techniques in general in different domains and more specifically in cloud brokerage.
2. A comprehensive study about cloud brokerage and decision support system methodologies.
3. Experiential insight about cloud providers and performance variations across different virtual machine instances

### 1.5.4 Thesis Organization

The remainder of the thesis is structured as follows:

*Chapter 2* provides an understanding of the problem space as well as a background overview of cloud brokerage and decision support systems. A detailed analysis of related decision support solutions is provided. In addition, the chapter gives an overview of using machine learning to achieve the goals mentioned in section 1.5, also exploring the state of art where machine learning methods are used in decision making. The chapter concludes by highlighting the potential of machine learning to be used for decision making integrated with cloud brokerage in a multi-cloud environment.

*Chapter 3* describes the architecture of Daleel for cloud instance selection.Following this, the chapter describes the key principles behind machine learning and also explores the core intelligence aspects of decision support systems. Finally, selected machine learning methods are explained along with their potential benefits.

*Chapter 4* provides an experimental evaluation of different learning strategies leading to the adoption of a set of approaches. The chapter also highlights a possible performance issue over training overhead. The chapter concludes with the final architecture of a generic learning model along with as assessment feasibility across different applications.

*Chapter 5* investigates a transfer learning technique to enhance the efficiency of an intelligent decision support system and reduce the training overhead in terms of time and cost. In particular, this chapter introduces a novel two-mode transfer learning scheme with the goal of achieving substantial reduction in this overhead. Following this, a detailed evaluation is carried out using two public cloud providers i.e, Amazon Web Services (AWS) and the Google cloud.

Finally, *Chapter 6* concludes this thesis highlighting the main contributions and detailing future work. Furthermore, this chapter revisits the research goals, showing where the questions are answered in the thesis.

Chapter 2

# Background & Related Work

Decision making in cloud environment is quite challenging due to the proliferation of service offerings, pricing models and technology standards. A customer entering the cloud market is overwhelmed with a host of difficult questions without much of a support for a decision support system. Moreover, there is no hard and fast rule for optimal selection of instance types that best suit the application needs and customer constraints.

The previous chapter has introduced key goals of this thesis, highlighting key limitations of existing decision support systems, along with the contributions and the adopted research methodology. The central tenet of this chapter is to investigate the current state of the art in decision support system, either offered as a service by cloud broker or just an independent effort in a multi-cloud environment. Moreover, the potential role of machine learning is also explored for developing an intelligent decision support system for a multi-cloud environment.

The upcoming sections cover the detail in the following manner, as shown in Figure 2.1. A quick recall of the problem domain is provided in Section 2.1. Section 2.2 defines cloud broker and classification of broker based offerings along with some of the brokerage examples. The role of decision support system and the current state of the arts are described in Section 2.3. This Section elaborates different methodologies involved in decision support system in the cloud or multi-cloud environment. The role of machine learning for developing decision support methodology is explored as well, and some supportive examples are stated in Section 2.4. The last section discusses the conclusion in view of limitations of existing approaches.

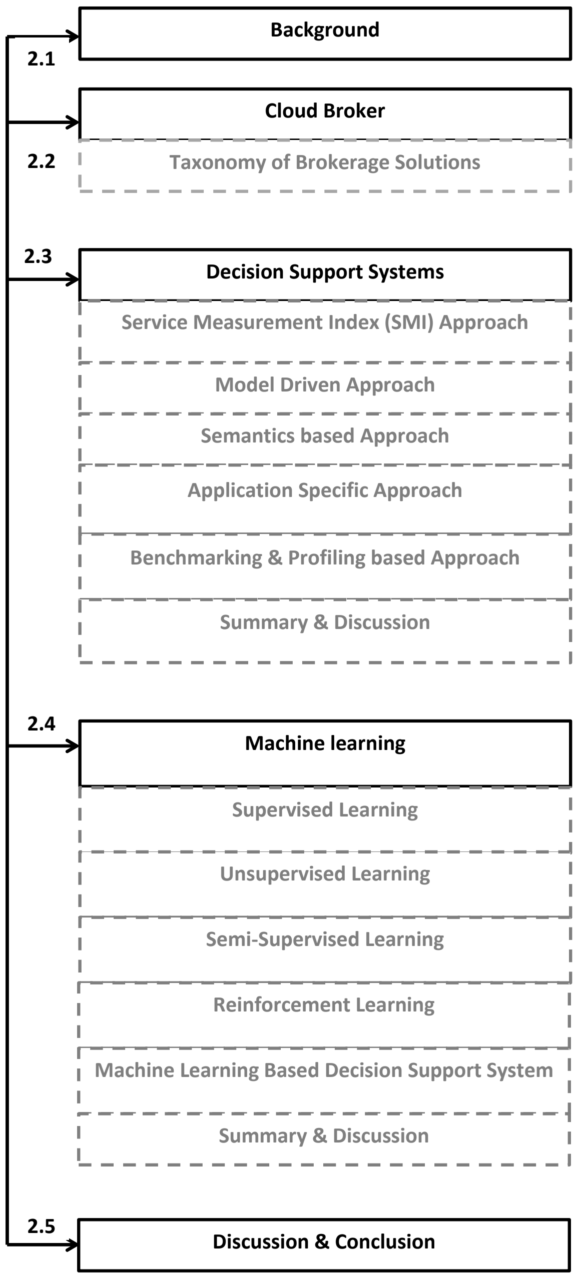


**Figure 2.1:** Chapter structure

## 2.1 Background

Cloud computing has opened up a world of new opportunities, not only for larger organizations but for small and medium sized businesses as well. As the dimensionality of cloud computing is increasing, it is facing many challenges such as energy efficiency, interoperability, resource utilisation, service provisioning, security, green computing, SLA management, heterogeneity and many more [32, 33, 34, 35, 36, 37, 8, 38]. Thus, researchers are striving to find best possible solutions.

One of the biggest challenge of cloud computing is inherent complexity in terms of different technologies, terminologies, services and interfaces. Every cloud provider is opting for different approaches of service offerings, pricing models for services and interfacing with its services. This variety is reflecting series of issues starting from vendor lock-in, portability to the performance comparison across the provider's offerings. The interoperability and portability are important for end-user investors as many of them do not want to stick their applications to one cloud provider only [14]. The cloud customers want to avoid the risk of being tied to one cloud provider to avail the option of application migration due to pricing and availing similar service with additional offers from some other cloud provider. The goal of interoperability and portability is to allow cloud customer to make best use of diverse offerings from cloud providers.

The cloud performance comparison is an important aspect of cloud and **Infrastructure as a Service** (IaaS) service selection for cloud customers. The **Infrastructure as a Service** (IaaS) selection is significantly important for the end-user as there is no straight method to compare the virtual machine performance within or across cloud providers. A wrong decision can lead to the financial as well as reduced application performance loss. A common practice by the user for selection of cloud provider is based on experience or reputation. On the other hand, virtual machines are selected simply by matching configuration details with the offered virtualisation service of that particular cloud provider. However, such selection criteria cannot be considered optimal in every case due to hidden uncertainties of cloud offerings such as scheduling algorithm, load balancer policies, co-location strategies, virtual to physical machine mapping rules, etc [39, 40]. In contrast to reputation based selection, one can explore all the possibilities regarding cloud provider and their offered infrastructure services as selection criteria. Considering the dimensionality of cloud providers along with the offered infrastructure services, the exploration exercise conducted by the cloud-user is not feasible in terms of time and cost. The selection criteria is not just a one-time task, it is an ongoing process till the end of application life cycle. Such activity becomes a hectic milestone at deployment or migration level. A new user entering the cloud market has to suffer from the cumbersome selection task and overwhelming thoughts of potential risk to wrong selection. Diversity of service offerings in terms of pricing model, functionality, virtual machine categories with various configuration options has raised the complexity of service comparison. There is no rule of thumb for transparent service comparison and selection under diverse conditions.

A cloud broker can help in resolving such issues by acting as an intermediate between the cloud provider and cloud consumer and offer a decision support system to assist the customer through the decision process. Machine learning assisted methods can enhance the potential role of decision support system by adding intelligence for application-driven decisions. An intelligent decision support system as a brokerage service can reduce the customer's efforts for optimal selection of infrastructure resources in a multi-cloud environment. This can lead to the satisfaction of application needs as well as user constraints in an optimal way. Here, the term "multi-cloud" denotes the usage of multiple independent clouds by a client or service [33, 41].

## 2.2 Cloud Broker

A cloud broker is an entity in a cloud ecosystem to manage the use, performance, and delivery of services along with negotiation of the relationship between the cloud provider and cloud consumer [42]. A number of tasks can be done by cloud brokers on behalf of customers like arbitrage, aggregation and integration/intermediation. Generally, we can define cloud brokerage solutions as either hosted or deployable. *Hosted services* are mostly commercial ones and are externally managed by the third party stakeholders. Such services are not transparent to the users and do not provide information of how the application is being provisioned. In contrast, *deployable services* are most commonly open source solutions and can be managed internally by a corporation or externally as grey-box service.

### 2.2.1 Taxonomy of Brokerage Solution

By looking through the literature at the term "broker" in a broad view of cloud and multi-cloud environment, we can highlight a list of services inclined with the brokerage solutions delivered by the broker itself. By definition, a cloud broker intends to offer different services considering its role between cloud-provider and cloud-user. However, cloud brokerage is still in its infancy and no single solution is reflecting full essence of brokerage system. We have identified and categorised some of the offered services into three level scheme of classification considering the intricacy of cloud broker role. Figure 2.2 shows an abstract list of offerings for these levels. Different state of the arts developed over time capture different aspects and offerings of cloud broker that are discussed in conjunction with classification levels. This three level taxonomy is considered a representative of literature knowledge to clarify the scope of cloud and multi-cloud management in brokerage domain. The intention behind this effort is to give our reader a clear picture about cloud brokerage based research and development efforts with a glimpse of problem domain and potential research focus.

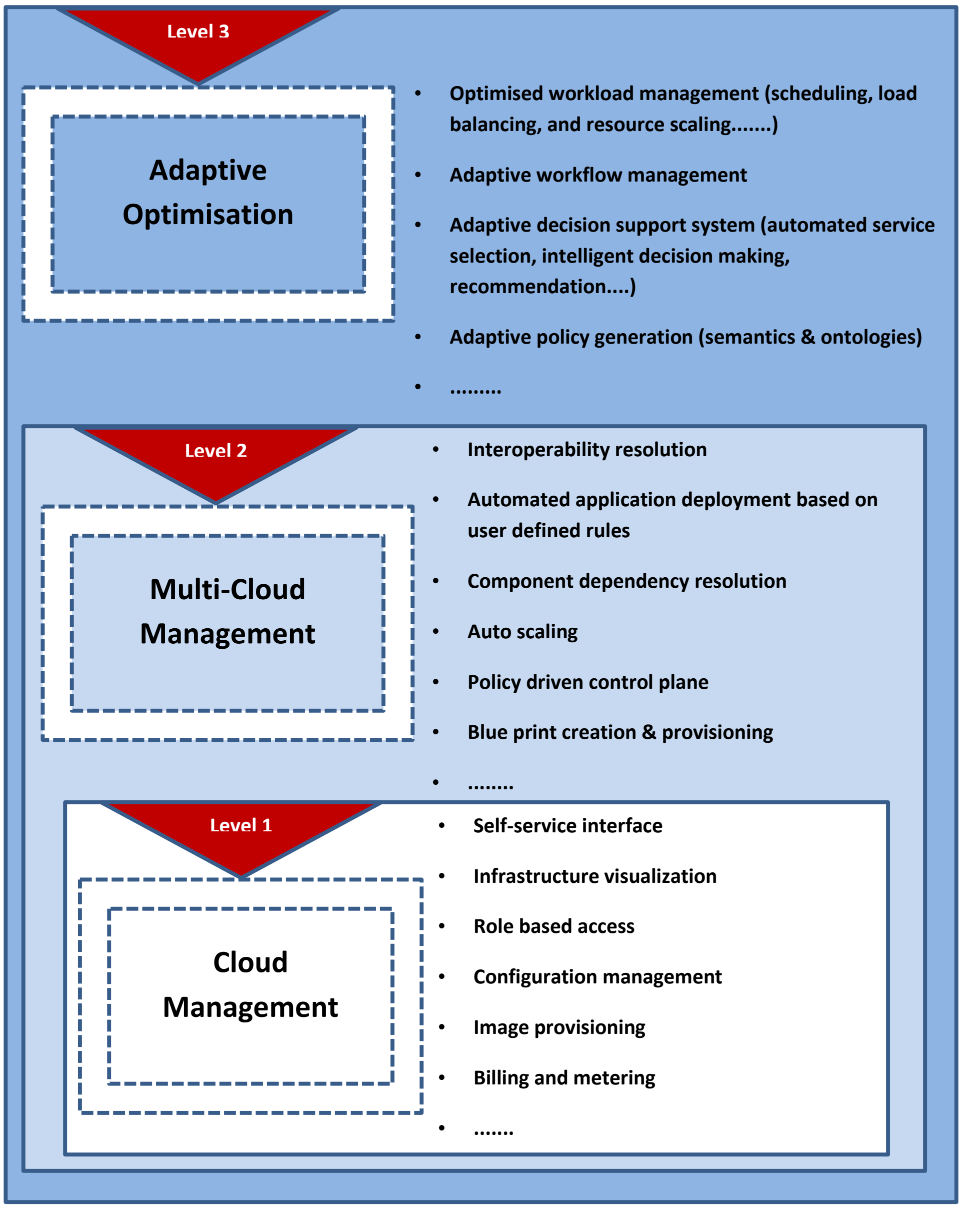

Level 3
Adaptive Optimisation
Optimised workload management (scheduling, load balancing, and resource scaling.......)
Adaptive workflow management
Adaptive decision support system (automated service selection, intelligent decision making, recommendation....)
Adaptive policy generation (semantics & ontologies)
.........
Level 2
Multi-Cloud Management
Interoperability resolution
Automated application deployment based on user defined rules
Component dependency resolution
Auto scaling
Policy driven control plane
Blue print creation & provisioning
........
Level 1
Cloud Management
Self-service interface
Infrastructure visualization
Role based access
Configuration management
Image provisioning
Billing and metering
.......

**Figure 2.2:** Three level classification scheme for cloud broker

*Level 1* refers to the cloud management platform. The offerings and list of important tools at this level are not considered as brokerage service, however, a necessary bridge to build smart and adaptive brokers. The services at this level include self-service interface, infrastructure visualization, role based access, configuration management, image provisioning, billing and metering. Configuration management tools like **chef** [43], **puppet** [44], **Ansible** [45], **CFEngine** [46], **SaltStack** [47], etc are used as supporting tools in this layer and layers ahead for similar tasks. *Level 2* refers to the provisioning of multi-cloud management solutions. This level contains services related to interoperability resolution, automated application deployment, dependency resolution and scaling management with the policy-driven control plane.

In the last couple of years, considerable developments have taken place in terms of solving the vendor lock-in and portability issues. The development of libraries/APIs is considered a principal effort to help reducing lock-in issue. Some useful libraries developed over time, to enhance interoperability, are discussed here. **Apache jclouds** [48] is an open source multi-cloud toolkit for Java to create an application that is portable across clouds. This library allows the full control of the cloud features to be used by the customer. **Apache Libcloud** [49] is a Python library and provides a unified API to interact with different cloud service providers. **Apache Deltacloud** [50] (moved to Apache Attic) is a REST-based API written in Ruby to interact with various cloud resources. **Simple Cloud API** [51] is a PHP library to access cloud application services offered by multiple vendors. It provides support for file and document storage and queue service of Amazon and Azure.

These libraries are facilitating interoperability and portability at different level of cloud services like virtual machine, storage, queue service, etc. Such solutions have opened up the venue for cloud brokers and cloud consumers to take full advantage of the multi-cloud environment. Various factors can lead to the use of multi-cloud environment like cost optimization; improve in QoS, customer specific constraints regarding storage or geographical locations, and sometimes to handle the peaks in demand of service. The commercial brokerage services as well as open source community has taken advantage of abstraction libraries to offer different multi-cloud management services/solutions. Some examples related to cloud and multi-cloud management are discussed here. **RishtScale** [15] offers a service to deploy and manage an application in different clouds. Users can manage the application on virtual machine of multiple clouds using the provided console. Users can add private cloud as well using the console. Scale up and down policies can be added as action by users and he can also specify the location of resources to be provisioned in case of scale-up policy. **Enstratus** [16] (on sale) is a unified management solution for deploying and managing enterprise-class applications on public and private clouds. **Standing Cloud** (part of App Direct now) [21]assist to deploy and manage applications on a variety of IaaS platforms without losing control or flexibility. **Kaavo** [52] offers a service to deploy and manage any application across multiple clouds. *IMOD* is Kaavo's core product for secure application deployment, scheduling and automation. This product is offered as a SaaS management solution as well as on premise solution for enterprise customers. As cloud computing evolves, the integration of cloud services becomes hassle for cloud consumer. **Brooklyn** [19] is an open source application management solution to deploy your application across multi-cloud environment. It also provides features for application monitoring and some base level scaling based on user defined policies. Similar to Rightscale, **Aeolus** [22] Blender intends to provide a solution for the automatic deployment and configuration of complex service-based distributed application. Blender is a software product maintained by Mandriva, based on the approach taken by the Aeolus project to overcome the limitations of using pre-configured images. Blender is an integration of three tools, Zephyrus, Metis and Armonic. The provisioning features of Aeolus are similar to Rishtscale and Brooklyn, feasible for multi-cloud application management rather automated decision making to assist customer for virtualization selection. **RightScale**, **Enstratus** and **Kavoo** are some examples of the commercial broker services. In contrast, **Brooklyn** and **Scalr** [20] are considered deployable solutions for the cloud user.

The examples discussed so far from cloud brokerage solutions are focusing more on resolving interoperability and providing solutions for cloud or multi-cloud application management solutions. These solutions are catering the need for automatic deployment, monitoring and defining the deployment and scaling requirements. These requirements are fed by the application-user in terms of specified configurations, targeted cloud provider along with basic scaling rules. In short, all decisions regarding application deployment are based on statically defined constraints. The most critical among these static constraints is the selection of suitable cloud provider. Further to that, the user has to be precise about mentioning the required configurations at virtual machine level.

*Level 3* in Figure 2.2, is considered an advanced level with a focus of adaptation and goal optimization. Researchers are applying enhanced methods like mathematical, analytical, machine learning and semantics to generate adaptive and optimised solutions for various goals related to cloud and multi-cloud management. Decision support system is part of this level that is a prime focus of our objective defined in chapter 1. The decisions could be related to the selection of cloud provider or service selection, particularly at the IaaS layer. **STRATOS** [23], **CELAR** [53], **MODACloud** [54], **mOSAIC** [37] and **OPTIMIS** are considered some of the initial efforts for providing decision making service in addition to multi-cloud management solutions. These state of the arts are discussed in details in Section 2.3. Level three is also considered a target point for machine learning methods to optimise the goals. Machine learning has proven its effectiveness in various aspects of distributed system approaches like scheduling, load balancing, etc. Very few efforts are seen in the literature as decision support strategies for multi-cloud environments. PuLSaR, Matrix and [55, 56] are considered some of these efforts to help customers by recommending cloud and service selection.

The next sections will highlight some of the potential efforts regarding decision support system that are either offered as cloud brokerage service or independent efforts. The state of the art in decision support system is categorised based on different methodologies as shown in Figure 2.3. The contributions and limitations are also discussed keeping in view the objective of this research work.

## 2.3 Decision Support Systems

The rapid increase in cloud service offerings has increased the opportunity for users to have redundant and replaceable services from multiple providers. As a result, the quality of service may improve and the customer can get even better performance at a reduced cost.

Users of Infrastructure as a Service (IaaS) level are faced with a composite decision:

1. Which provider she should choose?
2. What instance types(s) would provide her with the cost: performance ratio that suits her needs?
3. Does the time of day at which she requests these resources affect how her application runs?

No general rule can be applied to answer such subjective questions. Each customer application based requirements has to be considered carefully against the various cloud offerings before selection of resources. Further complications are manifested due to disparate pricing models across different cloud providers. The public cloud providers are offering the infrastructure services with personal defined standards and pricing options. For example, Amazon uses the term ECU what they refer as computation units to express the CPU capabilities of its various compute offerings while Rackspace defines CPU as a proportion of the physical host machine. In terms of pricing, Amazon charges for virtual machines on hourly basis, Microsoft azure bills on a per minute basis while Google charges a minimum of 10 minutes per virtual machine. Furthermore, the public cloud providers are offering their infrastructure service under self-defined categories based on the different configuration settings of virtual machines to cater specific needs of the application. The optimal selection of suitable instance

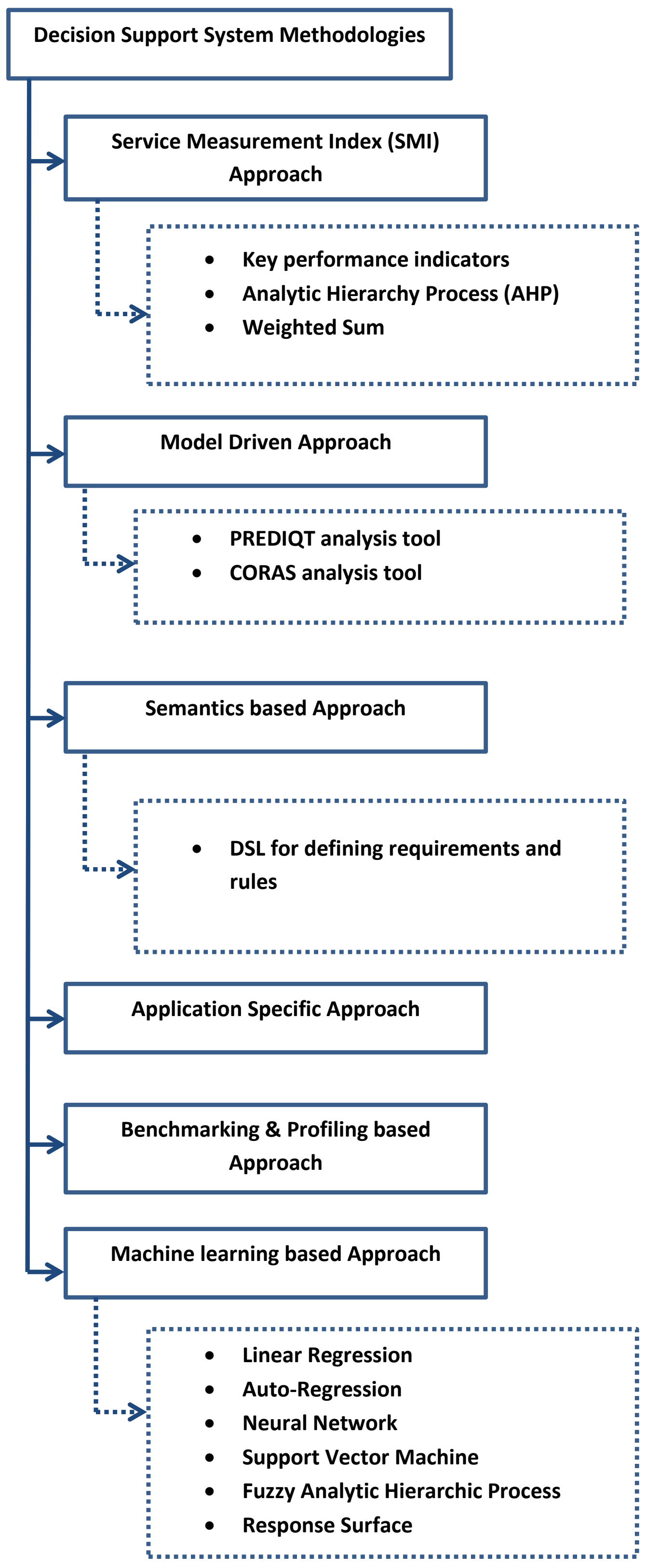

Decision Support System Methodologies
Service Measurement Index (SMI) Approach
Key performance indicators
Analytic Hierarchy Process (AHP)
Weighted Sum
Model Driven Approach
PREDIQT analysis tool
CORAS analysis tool
Semantics based Approach
DSL for defining requirements and rules
Application Specific Approach
Benchmarking & Profiling based Approach
Machine learning based Approach
Linear Regression
Auto-Regression
Neural Network
Support Vector Machine
Fuzzy Analytic Hierarchic Process
Response Surface

**Figure 2.3:** Categorical distribution Of decision support system methodologies

types from the range of infrastructure offerings is still a challenge for cloud users especially when user defined objectives has to be satisfied. So a cloud customer entering the market is overwhelmed by excessive hosting information without much knowledge of selection criteria.

An effective decision support system can help the customer by handling the bulk of information and ensuring the appropriate selection decision according to the mentioned requirements. This section discusses different approaches and methods for decision support systems. The offerings and limitations are critically evaluated keeping in view the research objective. These decision support solutions are explored with a viewpoint of brokerage in a multi-cloud environment.

### 2.3.1 Service Measurement Index Approach

The **Service Measurement Index** (SMI), developed by Cloud Service Measurement Initiative Consortium (CSMIC), is considered one of the approaches to compare different cloud offerings/services [57]. SMI is a set of Key Performance Indicators (KPI's) to provide standardized methods for measuring and comparing the services. These KPI's are categorized as accountability, agility, assurance, financial, performance, security and privacy, and usability. These seven characteristics are further divided into four or more attributes to capture all of its properties. These SMI indicators have been used by some of the commercial or open source broker based solutions for ranking different cloud providers and their offered services. Further, these ranking methods are used for decision making of cloud or service selection.

The work done by [58] for ranking of cloud computing services is considered one of the initial efforts for using SMI indicators. The proposed strategy measure all the KPIs of SMI and rank the cloud services based on the measuring results. An Analytical Hierarchical Process (AHP) based mechanism is proposed for the ranking mechanism. The proposed ranking is a three-phased process. Starting with the decomposition of a complex problem in a hierarchical structure to specify the interrelation among QoS attributes sub-attributes and alternative services. In the second phase, a relative property is calculated for each pair of QoS proceeding with pairwise comparison of cloud services for computing their local ranks as part of the third phase. The modeling of qualitative attributes is not in the scope of current work. Some other efforts are also done to define additional key performance indicators for SLAs and improved ranking methods [59, 60].

**STRATOS** [23] is a cloud broker service to facilitate the multi-criteria Resource Acquisition Decision (RAD). Three objectives are considered in the RAD decision: cost, vendor lock-in, and online provisioning. This methodology requires the application owner to specify the application requirements in terms of KPI. The KPI based requirements are then compared to all the provider's offerings to select the best acquisition. This selection acquisition is used at deployment time as well as at run time to fulfill the elasticity requirement. All the application specific details and objectives are defined in a Topology Descriptor File (TDF). SMI is used to measure cloud performance and the selection of resources is based on the defined details in TDF. Weighted sum method is used for objective optimization.

The main idea behind the **OPTIMIS** [36] project is to offer a toolkit for optimizing the whole service lifecycle, including service construction, deployment and operation on the basis of trust, risk, and eco-efficiency. In short, a suitable Infrastructure Provider (IP) is selected for Service Provider (SP). This project does not rely on the use of SMI, however, the measurement techniques for decision metrics are similar to the measurement of KPIs. The service life cycle starts with the service construction, where Service Provider (SP) prepares and configures the virtual machine images that constitute the service along with a specification of dependencies. As a next phase, most suitable Infrastructure Provider (IP) is selected for hosting the service. The selection process is based on the deployment cost along with other metrics including trust, risk, and energy impact. Reputation is considered a subjective measure of the perception of social network members regarding trust and risk analysis. One of the components of OPTIMIS is a deployment engine that is responsible for service packaging, discovery, and negotiation with a suitable cloud for hosting a particular service. The admission control component is responsible for analysing the current workload on infrastructure provider to carry on with the acceptance or rejection of service deployment request. After deployment, service and cloud optimizer components of OPTIMIS are responsible for monitoring the provisioning according to agreed SLAs.

***Analysis:*** To measure the long list of SMI indicators along with KPI requires proper understanding of cloud environment and fine grain application specific requirements. The user has to opt different methods for measuring the functional and non-functional based KPIs that may lead to monitoring the run time cloud environment with application specific fine-grained information like CPU, memory, bandwidth, etc. Past experiences are also counted as an input to some quantitative measurements that cannot be standardized. Moreover, consideration of reputation of the certain cloud provider to evaluate certain attributes like trust and risk can end up in a biased decision. In addition to that, virtual machines are selected by matching the user defined configuration requirements with the offered cloud infrastructure services. Such resource acquisition cannot be considered application-driven without knowing how the application will actually perform.

### 2.3.2 Model Driven Engineering Approach

Model driven engineering based solutions help application developers to model vendor-agnostic cloud services. A model-driven framework combined with different QoS based model-driven analysis can help in quantifying performance for the future. The **MODACloud** [54] project developed is a model driven approach for design and execution of applications on multiple clouds. It aims to provide a decision support system, an open source IDE, runtime environment for high-level design, semi-automatic code generation and automatic deployment on multi-cloud environments. A decision support method (DSM) [61] is presented to assist a cloud customer for selecting a cloud provider as well as service considering three QoS factors: risk, quality, and cost. This DSM is using the concepts of CORAS and PREDIQT to help to analyse these three QoS attributes. **PREDIQT** is a model-based quality assessment tool to predict the impacts of architectural design changes on system characteristics like performance, scalability, security, etc. **CORAS** is a model-based risk assessment tool and based on ISO 31000 risk management standards. It provides a customised language for threat and risk modeling and Unified Modelling Language (UML) is typically used to model the target of the analysis. The types of the proposed decision models are based on the modeling notations, languages and tools of CORAS and PREDIQT.

Similar to MODACloud, **PaaSage** [62] follows the principals of model-based software engineering. PaaSage is an open and integrated cloud management platform supporting design, development, optimization and deployment of existing and new applications. Cloud Modelling and Execution Language (CAMEL) is used by PaaSage to describe the application deployment model and scalability rules. The deployment starts with the translation of workflow description into a CAMEL model which is then consumed by another PaaSage component to generate a deployment plan. Finally, the deployment plan is passed to the execution component of PaaSage to perform the actual deployment. Application auto scaling is also managed by the execution component of PaaSage based on the rules mentioned against the workflow execution stage.

**ARTIST** [63] is another model-driven engineering based project to be used for migration of legacy applications on the cloud. The methodology is a three phase process starting with feasibility analysis, modernization of software and validation in post-migration phase. The decisions are mostly related to the business level, where a feasibility analysis is done to foresee the investment risk. However, both technical and business aspects of migration are covered to help the user in customizing according to application needs. The outcome of this project is a toolbox for the migration, modernization, and cloudification of the legacy application. The core of the offered software suite is the Methodology Process Tool for customisation and instantiation of the generic methodology on the pre-migration analysis results.

**Analysis:** Model-driven engineering based solutions are heavily dependent on the fine-grained information from domain experts, analysts and decision makers to get complete knowledge of business models and company strategies. A designer must be aware of the impact of decisions, alternative decisions, actor interactions, dependencies, and processes while designing the workflows and architectural models. Such processes require sufficient time to follow the model based design principals. In the case of DSM-MODACloud, there is no clear picture of how the cloud service measurements will be collected and which parameters to consider.

### 2.3.3 Semantic Approach

Semantic technology helps in representing resources, domain concepts and rules with the help of ontologies. In this way, application requirements and service specifications can be defined in a vendor independent way raising the abstraction for the user. **mOSAIC** [37] is an open source PaaS programming interface for the development and deployment of multi-cloud based applications. mOSAIC has an assumption about the deployed application that it is component based and dependencies for both communication and data are defined explicitly among components. Furthermore, application architecture must be service oriented and only the mOSAIC API should be used for inter-component communication. Resource Brokering is one of the components of mOSAIC to act as a mediator between client and cloud provider. The mediation involves the responsibility of cost optimization and performance maximization. The mOSAIC platform assists the user for automatic provisioning without direct user involvement. However, the SLA based performance indicators at application and component level have to be defined by the user. mOSAIC assists the customer at PaaS level in order to combine services from different clouds and to provide a new service on top of existing ones (and provide transparency of multiple clouds).

In the context of the mOSAIC project, a knowledgebase is developed representing resource and domain concepts by means of Semantic Web Ontologies and inference rules. The proposed ontology is focusing on the IaaS layer for resource provisioning. This knowledgebase (support tool) is comprised of two components, Semantic Engine and Cloud Agency. Semantic Engine (a prototype tool) is responsible for helping the user to abstract the requirements in vendor independent way starting from application requirements or from specific vendor resources. A Cloud Agency compares the different offers of providers with the user proposal and retrieves the best offer. Cloud Agency is a Multi-Agent System conceived for provisioning by negotiation, monitoring, and reconfiguration of acquired resources. However, the proposed approach is taking advantage of inference rules and trying to overlap the gap between high-level requirements and technical requirements.

**Cloud4SOA** [25] is a multi-PaaS approach to semantically interconnect heterogeneous PaaS offerings across different cloud providers that share the same technology. This is a broker based solution for the PaaS layer so the developers can select, deploy and manage their applications on PaaS offerings and can switch between platforms without re-architecting the original application. The ontology-based semantic matchmaking capabilities provided by Cloud4SOA establishes a set of abstraction among different PaaS offerings to support seamless deployment and management of application across different cloud platforms. Cloud4SOA is based on five layer architecture to support Service Oriented Architecture (SOA), user-centric focus, PaaS Semantic Interoperability Framework (PSIF) and management of the cloud-based application. The PaaS Recommendation module in SOA layer is responsible for suggesting for the best match of PaaS offering. The semantic profiles of application and PaaS offerings define the degree of their relation to helping in matchmaking.

***Analysis:*** mOSAIC and Cloud4SOA are multi-PaaS semantic based management and governance solution for the multi-cloud environment. Use of ontologies and semantic technology is an effective way of abstracting the complexities from cloud user. However, it comes with an on-going follow-up process. For example, to present the actual offering details along with some new addition, the semantic database has to be updated all the time. On the other hand, Cloud4SOA offers the PaaS Semantic Interoperability Framework (PSIF) for the formal presentation of information regarding PaaS offerings, application and user profile. So, all the offerings and details must be defined in a standard way to be used in this system.

### 2.3.4 Benchmarking & Profiling Approach

Due to the diversity in cloud computing infrastructure services, it is very difficult for the user to choose which virtual machine has the maximum performance capability to be selected for deploying the application. Sometimes, the wrong choice of selection ends up with under-performance of application or increase in resource cost. One of the methods to help categorize the virtual machines is to benchmark the performance. Mostly the benchmarking is performed independently of the application which is not an appropriate selection and categorization method as the heterogeneous virtual machine would have varying performance capability for various standards like CPU utilization or memory utilization etc which might not be a good choice for some other application.

A six step benchmarking methodology is proposed by [29]. In this strategy, different virtual machines are grouped (memory & process group, computation group, local communication group, and storage group) together based on the similarity of VM attributes. The attributes are normalized to rank the VM performance within each group. For each input application, weights are assigned to the respective groups by the experts. In this way, a relevancy of application is shown with the group using a value ranging from 0-5. Application experts indicate the relevance of application to the four groups (computation, memory, processor and storage) by assigning weights to each group. Scores are calculated by multiplying the weight and normalized value of the group and virtual machines are ranked accordingly. A high ranked virtual machine can be selected now to maximize the application performance.

**CloudCmp** [26] strategically measures the performance for a range of different cloud services for multiple cloud providers. These services include elastic compute cluster, persistent storage, and intra-cloud networking. The performance of these services is evaluated using different benchmarks and metrics e.g., the speed of CPU, memory and disk I/O, storage service response time, scaling latency, network latency, available bandwidth etc. The main goal of the work is to run the benchmarks to measure above mentioned services on four providers and then use the results for real-time application deployment. In this way there is no need to run the application on each cloud provider, instead use the benchmark results to select the provider and resources according to application type. Java based benchmark tasks from SPECjvm2008 are used to evaluate the computation metrics which includes benchmark completion time, cost per benchmark and scaling latency. Storage metrics evaluates response time, throughput, cost and time to consistency. The Network metrics are measured using standard tools like iperf and ping.

**Profiling-as-a-Service (PraaS)** [64] approach is an adaptive instrumentation strategy to collect profiling information of running application on a cloud while achieving the defined QoS. This service intends to provide help to perform a trade-off between performance, cost, and accuracy of profiling data. In terms of accuracy, application architects can define expected QoS and constraints to achieve a certain level of accuracy. The policies are then uploaded to the PraaS service and instrumented application is deployed for resource usage monitoring.This service is tested for an open source web-based application. The current implementation supports profiling in three modes, no profiling, partial profiling and full profiling that can be applied to target VM instance of Windows Azure. The main focus for profiling is the resource utilization (CPU usage and data exchange) by different components of applications.

***Analysis:*** The ranking method depends on the assigned weights by experts and so requires fine-grained application specification and requirement knowledge. In contrast, the benchmarking based decision making is considered a useful solution. However, such decisions cannot be considered application-driven as the behavioral change of application and cloud resources are ignored. IaaS offerings are black-box for the user; the different software and hardware based policies at virtualization level can cause uncertainty about decision outcome. Profiling methods are considered effective capturing irregularities about application behavior. This is very evident that by increasing profiled data ratio, better results could be derived, though at a cost of time and budget. Such solutions show no relevance to broker based system though considered substantial efforts for decision making.

### 2.3.5 Application Specific Approach

Some research efforts regarding decision making for resource selection are based on application specific solutions. These solutions may not come with the assumption of being part of a brokerage system or multi-cloud environment. Such solutions are also considered domain specific decision support methods and may not be performing effectively in different domains.

**Conductor** [65] is an approach to automatically manage cloud resources to satisfy user specific goals such as cost or completion time. The solution is restricted to MapReduce computations and the resource allocation problem is addressed using linear programming. It is quite feasible to use linear programming to build a generic model for MapReduce computations due to the availability of predefined data flow patterns. However, the same model might not be a viable choice for other kinds of applications.

To find out the suitable hardware and software configuration for an application is non-trivial, especially with customer specific constraints regarding budget, performance and time. A common practice is to follow the past experience for such decisions. **[55]** has applied the concept of Response surface ( a statistical method) for selection of hardware and software resources for MapReduce application. **Response Surface** model captures the performance change of application on different setups of hardware and software. In this methodology, a Response Surface is build using a range of exploratory variables including hardware platform, network, and software configuration. The parameter values can be collected from simulation to populate response service.

***Analysis:*** Simulation-based data can be collected easily in a faster way in a controlled environment; however it cannot represent the actual behavior of cloud environment on any application. Mathematical or statistical methods are considered a good choice to understand and explore the relationship among data attributes. This can lead to learning a system behavior. Such models can be easily used under enhanced machine learning methods.

### 2.3.6 Summary & Discussion

This section has investigated the related work to decision support system for cloud brokers in a multi-cloud environment and identified methodologies for decision making. The state of the art is summarised in a Table 2.1 to quickly evaluate the contribution. This table is composed of 9 columns where the first column represents the state of the art. Second column defines the cloud service layer where the decision support methodology is applied to Infrastructure as a Service layer or Platform as a Service layer. Third column defines the decision support method opted by the state of the art. Fourth column represents the decision making criteria in terms of quality of service attributes or user defined attributes. Fifth column declares the categorical name of the decision support methodology by which state of the arts is categorized. Some additional dimension of summarised comparison is added in the last four columns to indicate whether approaches are designed for single or multi-cloud environments; how much information has to be provided by the user; and how easy it is for each approach to respond to changes in the cloud market. Last column reports the application architecture approach is tested on or provide support for. In the table, fine-grained information refers to the low level configuration specific detail like number of CPU cores, CPU speed, RAM, storage, etc. On the other hand, coarse-grained information represents a high level set of information in the form of threshold levels to define QoS like high performance or low cost, etc.

RightScale, Aelous, Kavoo, Brooklyn and Scalr are providing means of service offering for control and administration of distributed application deployed over different clouds, as stated in Section 2.2. These solutions help for automated deployment of cloud or multi-cloud application and the user is responsible for providing the choice for targeted cloud and virtual machine configurations. Optimis, mOSAIC, Cloud4SOA are offering platforms for management and governance of cloud services (deployable service for multi-cloud). Reputation is considered a subjective measure of trust and risk analysis in Optimis and STRATOS (Section 2.3.1). mOSAIC and Cloud4SOA also provide semantic-based solutions for matching of PaaS which requires a standard way of getting the requirements and matchmaking (Section 2.3.3). MODACloud, ARTIST, and PaaSage are considered Model-driven engineering solutions for multi-cloud and use analytical methods for decision making and requires a lot of information from domain experts, analysts and decision makers belong to business or organisation (Section 2.3.2). Ranking or matchmaking based methods cover fine grained requirements, however, do not consider how an application will perform on different cloud setups.

Benchmarking or run-time application profiling is an effective way for monitoring the service level objectives as well as provisioning of required resources (Section 2.3.4). Application profiling data is useful to capture performance change and to detect any implementation error. However, it comes with an additional overhead cost. Offline profiling can be used to reduce the overhead cost and to observe performance change in a controlled environment in order to define policies to apply in an on-line manner. This strategy sometimes is not suitable for the cloud environment where many unknown factors can change the application specific QoS. Utilisation of existing data traces is also used in application specific approach, where performance change is captured using mathematical and statistical models (Section 2.3.5).

Table 2.1: Summary: Decision Support System Methodologies

| State of the Art | Layer | Decision Support Method | Decision Criteria | Category | Supported Cloud | Level of User Specified Info. | Response to Changes | Supporting Apps. |
|---|---|---|---|---|---|---|---|---|
| [58, 59, 60] | IaaS/PaaS | KPI, SMI + AHP | Ranking of service | Service Measurement Index | Multi | Fine-grained | Re-calculate ranking with updated data | All (in theory) |
| STRATOS [23] | IaaS | SMI + Weighted sum | Cost, Lock-in | Service Measurement Index | Multi | Fine-grained | May need to re-adjust weights | Web benchmarks |
| PaaSage [62] | IaaS/PaaS | User defined rules using DSL | User Specified | Model Driven Engineering | Multi | Fine-grained | may need to re-evaluate selection after adding new rules, update ontological concepts | Scientific applications |
| ARTIST [63] | IaaS | Analysis of Legacy Applications | User Specified | Model Driven Engineering | Single | Fine-grained | Re-design business model or update ontological concepts | legacy apps. |
| OPTIMIS [36] | IaaS/PaaS | User preference + Reputation | Trust, Risk, Energy Efficiency | Management & Governance | Multi | Fine-grained | Update toolkit to accommodate new APIs | Not specified |
| mOSAIC [37] | IaaS/PaaS | Semantic Ontologies | User defined application and PaaS profile matching | Semantic + Management & Governance | Multi | Fine-grained | Update semantics and ontological model | Not specified |
| MODACloud [54] | IaaS/PaaS | Analysis using CORAS & PREDIQT tools | Risk, Quality, Cost | Model Driven Engineering | Multi | Fine-grained | May need to re-evaluate output | Distributed and standalone apps. |
| Cloud4SOA[25] | PaaS | Semantic Ontologies | Not mentioned | Semantic + Management & Governance | Multi | Fine-grained | Update semantics and ontological model | Business apps. |
| Conductor [65] | IaaS | Linear Programming | Cost, Completion Time | Application Specific | Hybrid | Coarse-grained | May need parameter tuning or model tweaking | MapReduce |
| Cloudcmp [26] | IaaS | Benchmarking | Computation, Memory, Processor, Storage | Benchmarking & Profiling | Multi | Coarse-grained | May require model update | Parallel scientific, E-commerce, latency sensitive |
| [55] | IaaS | Response surface | Hardware & Software specific | Application Specific + Machine Learning | Single | Coarse-grained | May require model tweaking | MapReduce |
| Cloudbench [29] | IaaS | Benchmarking & Weight Assignment | Computation, Storage, Network | Benchmarking & Profiling | Multi | Coarse-grained | May require new data for re-evaluation and weight adjustments | CPU, memory, network intensive benchmarks |
| Profiling as a Service [64] | IaaS | Application Profiling | CPU Utilisation & Data Exchange | Benchmarking & Profiling | Single | Coarse-grained | Re-profiling and model update | Web applications |

Statistical and mathematical methods are considered quite effective for defining the underlying relationships among data attributes. These methods are also linked with statistical machine learning, where the captured relationships are utilised in deriving adaptive and robust learning models for a given domain. This fortifies towards the argument of prospective use of machine learning for decision making. We will try to explore if machine learning can play a vital role to overcome many of the shortcomings in existing decision support methods. Limitations of existing state of the arts are summarised in different key questions in order to evaluate if machine learning can help answer these.

1. How can we use machine learning for application-driven acquisition decisions without much intervention of human choices? Particularly, the decisions related to virtual machine selection at IaaS layer.

2. Can we rely on machine learning methods to capture application behavior on different deployment environments?

3. Are machine learning methods useful in deriving inference relations between the data instances where some uncertainties are generated due to the black-box nature of cloud provisioning?

4. Can we integrate learning based decision support system with brokerage architecture?

5. Can we design and develop learning based solutions that are viable across multi-cloud environment?

The next section targets the evaluation of machine learning for decision support system keeping in view all the raised key questions.

## 2.4 Machine Learning

Machine learning is a branch of artificial intelligence, which helps a system learn from past experiences - or the so-called datasets or training sequences. about the construction and study of systems that can learn from data. Tom Mitchel [66] defined a learning problem as: “A Computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”

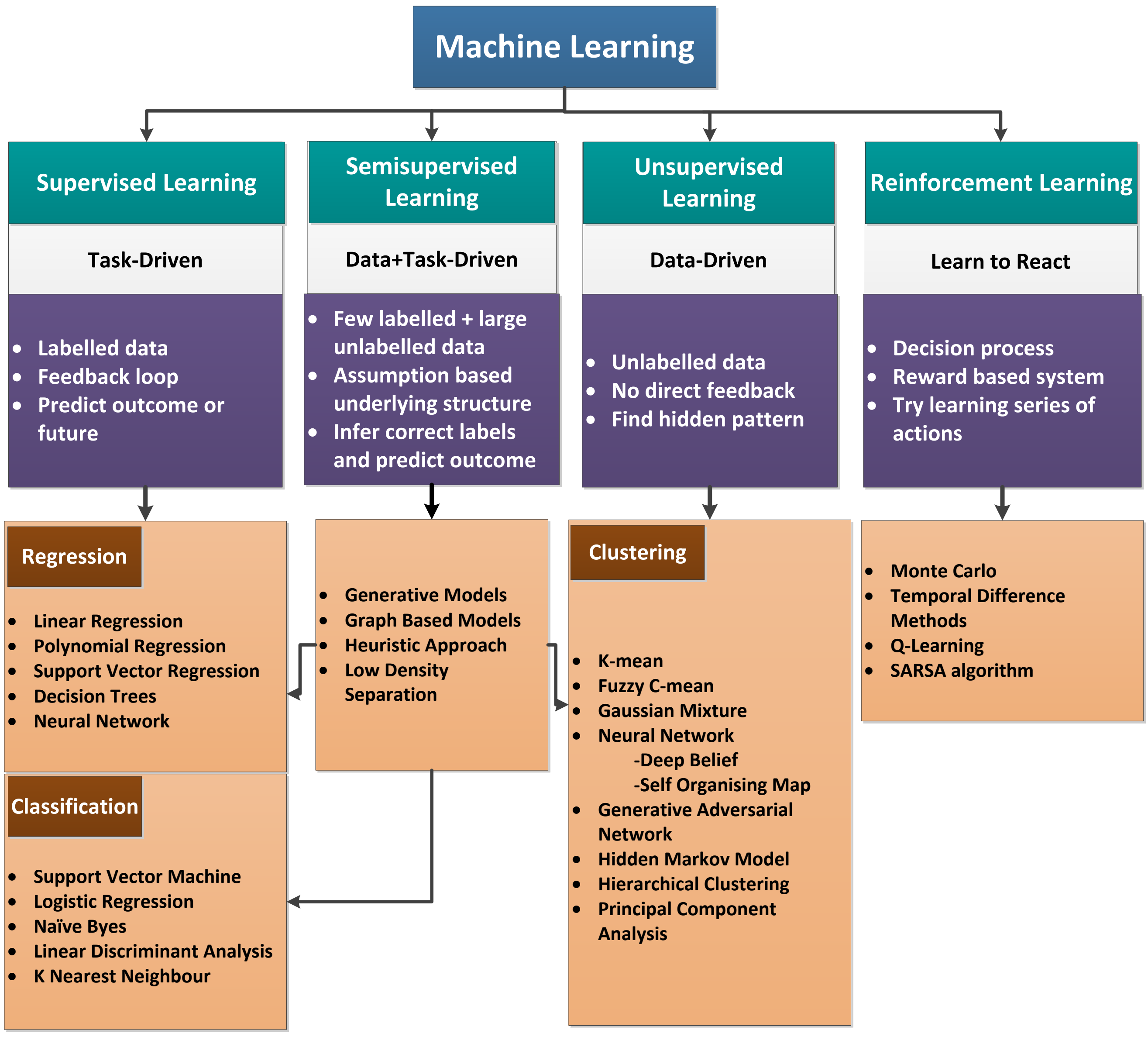


**Figure 2.4:** Classifications of machine learning with specified characteristics (purple box) of each category along with representative examples of algorithms.

## 2.4.1 Machine Learning Classification

Machine Learning can be divided into 4 categories, 1) Supervised learning [69] 2) Unsupervised learning [70] 3) Semi-supervised learning [67] [68] 4) Reinforcement learning [71] as shown in Figure 2.4. This figure explains high-level characteristics of each of the category along with some examples of machine learning methods.

1. **Supervised Learning**

   Supervised learning uses known data sets (labelled data) as training data for making predictions. The training data includes input data as well as response values. Explicitly, a supervised learning algorithm builds a model based on the training data, which predicts future response values for new data set, such as weather forecast to predict temperature or precipitation value.

   Supervised learning can be divided into two categories: *Classification* and *Regression* as shown in Figure 2.4. *Classification* is a way of learning from categorical response values, where data can be separated into specifically labelled classes. In other words, it can be classified as a discrete-valued output method, for example, to detect if the cancer is benign or not. By contrast, *Regression* algorithm is used for predicting continuous output response, for instance, predict a house price, and temperature prediction for the weather forecast. Some of the learning methods under *Classification* and *Regression* categories are listed in Figure 2.4.

   A workflow model of supervised learning is explained in Figure 2.5. A process of model generation starts with feature extraction from available labelled data. Extracted features are then used to fit a model which is then trained on a training data set in an iterative fashion to update the model. To check the accuracy of data, the model is tested on test data set and if satisfied then used for future prediction.

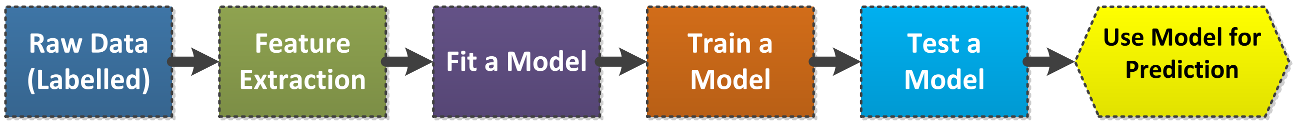


**Figure 2.5:** Supervised learning work-flow

2. **Unsupervised Learning**

   Unsupervised learning is a technique for finding hidden patterns or intrinsic structures in the data. It is used to draw inferences from datasets, which consists of input data without labelled responses. *Clustering* is a common technique for unsupervised learning, which aims at finding patterns or similarities using the Euclidean or the probabilistic distance. This technique is widely used in bioinformatics, computer vision, and medical imaging. Some representative algorithms for clustering are listed in Figure 2.4. A workflow model of unsupervised learning is also explained in Figure 2.6. A process of model generation starts with feature extraction from unlabelled data.

Extracted features are then used to group subsets of data into clusters and later on used to predict labels of new data.

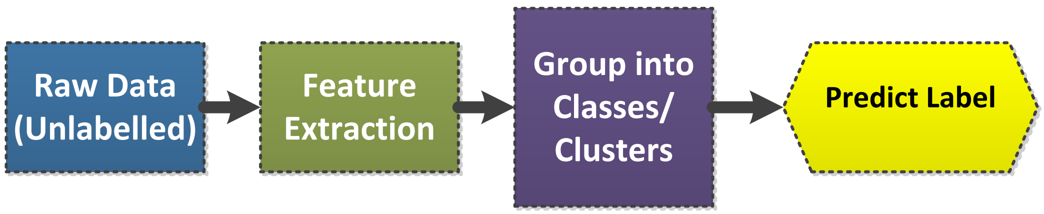


**Figure 2.6:** Unsupervised learning workflow

3. **Semi-Supervised Learning**

   Semi-supervised learning falls between the supervised and the unsupervised learning, because it considers labelled as well as unlabelled data for training. Some of the semi-supervised learning models are self-training models, mixture models, co-training and multi-view learning, graph-based methods, and semi-supervised support vector machines. Furthermore, since semi-supervised learning makes assumptions about the distribution of unlabelled data, for example smoothness assumption, cluster assumption, and manifold assumption, so it is a form of transductive and inductive learning.

   A workflow model of semi-supervised learning is explained in Figure 2.7. A process of model generation starts with feature extraction from few labelled and a large amount of unlabelled data. Extracted features are then used to fit a model based on initial assumptions about the underlying relationship of features with the response variable. Learned model is then used to infer correct labels/values for the unlabeled data and tested on test data set. If the model is performing well on test data then the model is further used to predict future values. Most of the learning algorithms defined for supervised and unsupervised learning can also be used under semi-supervised learning approach.

4. **Reinforcement Learning (RL)**

   RL aims for mapping situations to the actions so as to maximize the reward. RL is similar to behavioural psychology. Specifically, RL is a trial and error learning technique, which aims to strike a reasonable trade-off between exploration and exploitation- exploitation is when an agent

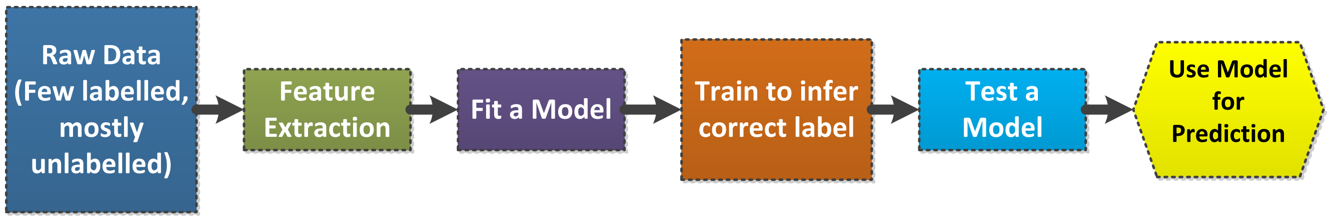

Raw Data (Few labelled, mostly unlabelled)
Feature Extraction
Fit a Model
Train to infer correct label
Test a Model
Use Model for Prediction

**Figure 2.7:** Semi-supervised learning workflow

uses its past experience, while exploration is to make a better action selection in future. On the success of an action, the agent receives a numerical reward. Hence, the agent tries to learn and select those actions, which would maximize its accumulated reward over time. A workflow model of RL is explained in Figure 2.8. Agent learns to react to some action by trial and error process of applying actions on the environment and analysing the result of action by reviewing next state as well as reward value. This type of learning is mostly applied in the robotics field. Some commonly known algorithms under this category are listed in Figure 2.4

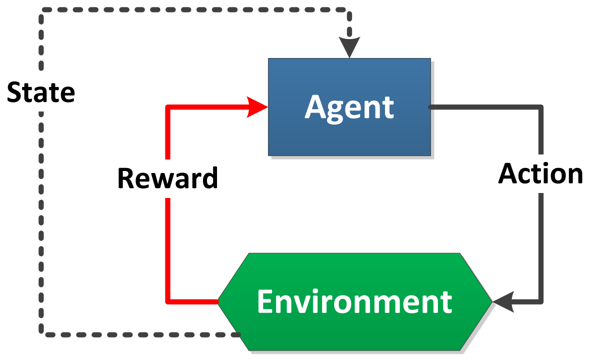


**Figure 2.8:** Reinforcement learning work-flow

### 2.4.2 Machine Learning for Decision Support System

Machine learning is not only contributing enormously to various areas, for example feature and speech recognition, bioinformatics and robotics but also performing quite well in resolving complex problems of cloud computing.

Machine learning has proved its effectiveness in various aspects of cloud computing environment like scheduling, load balancing and resource scaling, by forecasting the future needs. Machine learning also helps in deriving suitable decision models for complex application scenarios. Some potential research efforts are discussed here to explore the scope of machine learning in this domain. Furthermore, machine learning based decision support methods are also investigated in the context of multi-cloud brokerage solutions. In this regard, different machine learning algorithms are also highlighted to indicate their effectiveness for prediction methods.

This work[**72**] is considered one of the initial efforts for dynamic resource scaling in the cloud. Three platform-agnostic algorithms are analysed under the defined objective. One of the three algorithms is developed by RishtScale, while the two others predict system load using linear regression and autoregression of level 1. The Regression analysis is a statistical process for estimating relationship among variables of the dataset (scalar dependent variable y and one or more explanatory variables X). This relationship can be linear in nature if the relationship is modelled using linear predictor function. Autoregression is a stochastic process used to estimate future values based on weighted sum of past values. Furthermore, the authors' also established a scoring metric based on availability and cost for measuring the effectiveness and efficiency of these algorithms. Here, availability is counted by considering the number of dropped requests out of a total number of requests. The results demonstrate that linear regression is considered more susceptible to small fluctuations in the generated load. By contrast, autoregression shows less sensitivity to load fluctuations. In addition to that autoregression is far more reactive than RisghtScale algorithm.

**Caron et al.** [**73**] targets resource scaling by identifying the patterns of past incidences of short-term workload and matching with current occurrences, which is similar to string matching. Furthermore, Knuth-Morris-Pratt (KMP) algorithms is used to identify the similarities in the past and the current data. Historical data about CPU utilization is used as target pattern and a unit time of 100 seconds is fixed to make chunks of data for matching. This process is very time-consuming as the current pattern has to be matched with loads of historical patterns until the match is found. Another limitation is the time unit used for capturing data traces, which is not feasible in case of cloud computing environment where pricing schemes are different. For example, Amazon charges on hourly basis, so any delay in the decision can increase the utilization cost.

The objective of this work [**74**] is to provision resources ahead of time before these are actually required. Future demands are predicted earlier using different machine learning methods. Three quality of service attributes are considered as input to the prediction methods, the attributes are response time, throughput and CPU utilization. Linear Regression (LR), Support Vector Machine (SVM) and Neural Network (NN) are used as prediction methods. SVM is a supervised learning model with associated learning algorithms to analyse data used for regression and classification. The NN approach is inspired by the way a biological brain solves the problem where a large number of neurones are gathered in a cluster and attached with a central point. Artificial Neural Network (ANN) model shows interconnections between neurons in different layers of each system which defines interconnection patterns between different layers. The learning process updates the weight of interconnection based on the input and then activates the function to convert the neuron's weighted input to its output activation.

Applications relying on dynamic autoscaling techniques may not be capable of handling a sudden traffic surge resulting from special offers or events, and hence, turn out to be low in performance.This work **[75]** focuses on the limitation of reactive dynamic auto-scaling approach and the use of empirical data for adaptive resource provisioning. Similar to [74], this approach also targeted the predictive resource provisioning for web server using Neural Network (NN) and Linear Regression (LR) methods though using benchmark on Amazon EC2 to collect data for training and testing. This work also uses the sliding window method and tries to capture the workload patterns for forecasting. The efficiency of prediction model depends on the workload patterns only. This work also targets the limitation of [73] by reducing the unit time of 100 seconds to 60 seconds for data log in order to be reactive according to billing time for Amazon EC2.

**Jim [76]** has developed a neural network based framework that learns from actual data to model the performance of power plant. This work targets the application of machine learning for power optimization of Google data center. The machine learning based power performance model predicts the power usage in Google data centers and results in improved energy efficiency.

This work **[77]** targets the resource management decisions in cloud computing using machine learning. Support vector regression is used as a prediction method to estimate the response time for designing the resource allocation strategy. A Genetic Algorithm (GA) based resource dispatch mechanism is proposed for the relocation of resources. The resource dispatch mechanism aims at complementing the SLA between virtual machine operators and cloud service providers by effectively utilizing the resources and maintaining the desirable performance at cloud level.

***Analysis:*** The above-mentioned state-of-the art are applying different machine learning methods to achieve the goals of resource provisioning, auto-scaling, and power optimisation. Some of the work has applied the methodology to local virtual machines or simulated data that cannot reflect the actual domain of cloud or multi-cloud environment. Some specific applications are targeted as problem domains that cannot lead to generalised methods to be applied to a wide variety of application set. These are closer to application specific prediction and decision making. The core focus of these research methods is the prediction of upcoming workload to configure virtual machine ahead of time. So the decisions are related with "When" to provision rather than "How" and "Which" type of virtual machine to select. [77] and [76] are using machine learning for effective utilisation of resources and power consumption from the view of the cloud provider, which is slightly different from our core objective of assisting the customer.

Provisioning of intelligent decision support systems as brokerage solutions in a multi-cloud environment are very few in numbers. We will critically explore these solutions in next section to strengthen our argument about the potential domain to be explored.

### 2.4.3 Machine Learning for Cloud and VM Selection

Apart from these state-of-the art, some research efforts are started to surface for cloud and resource selection (hardware & Software) or virtual machine selection. These solutions are either offered as a generic recommender for service selection or application specific resource management in a distributed clustered environment. These selection methods are based on the use of machine learning techniques in various ways that are described in detail ahead.

**Matrix** [30] is a performance and resource management system based on machine learning methods. It uses clustering models with probability estimates to predict the performance of new workload on a different virtual machine. In this regard, Matrix constructs performance models for different workloads, while new workload is classified using the support vector clustering technique. Matrix is capable of recommending a VM, which is good in performance and incurs minimal cost. Support vector machine with different kernel functions is used to find optimized resource allocation. Represented applications or workloads are selected from some widely used benchmark suites e.g., FileBench, SysBench, SPEC2006, PARSEC and Cloud9. These benchmarks provide a diverse range of workloads to cover CPU intensive, IO intensive and memory intensive application types.

**Preference based cloud service recommender (PuLSaR)** [27] is a multi-criteria decision-making approach, which offers optimization as a brokerage service. Service evaluations are done using the comparison of qualitative and quantitative characteristics of cloud offerings. The author considers the use of imprecise and precise metrics, which are more meaningful for characterizing and ranking a cloud service. This recommender system uses the notion of imprecise metrics along with the precise one for capturing the fuzzy or linguistics, which are required by the customer. This approach is an extension of the SMI model, which now includes a new top-level attribute called Reputation, while some existing attributes like robustness, support and monitoring, are re-adjusted as a second and third level imprecise attributes. Based on this enhanced model the service KPI and user requirements can be fuzzy numbers and intervals. Furthermore, fuzzy Analytical Hierarchical Process (AHP) is used for comparison and ranking. Explicitly, AHP is a structural technique for analysing and organising complex decisions using mathematical and psychology-based methods.

This work **[56]** has presented twofold contribution. A CSP indexing technique is proposed to efficiently manage a large number of cloud service providers. The CSP-index is developed using the B+-tree data structure technique. This technique facilitates easy integration of new index structure with the existing system. The data structure includes ten property values for each service provider. The ten values stored as properties for each service provider, which are related to Quality of service, measuring units, pricing units, security, and virtual machine configuration. The focal point of using the properties is to quantify relationship among the service providers. Service providers are clustered based on the encoded index key. K-mean algorithm is applied for clustering, while iDistance is used to quantify similarity between points. Consequently, the same type of service providers will be clustered together. Once a customer enters his requirements, the requirement is transformed into a query to search for a match from cluster.

Response surface **[55]** is used as a statistical machine learning methodology for selecting suitable hardware and software resources for the MapReduce-based application. Alongside to the Polynomial Regression approach, Response Surface methodology explores the relationship between explanatory variables and one or more response (output) variable. In this methodology, a Response Surface is built using a range of exploratory variables including hardware platform, network, and software configuration. The parameter values for populating response surface can be collected from simulation. Simulation-based data can be easily recorded in a controlled environment; however, it does not portray the actual behavior of application on cloud environment due to its black-box nature.

***Analysis:*** The afore-mentioned state-of-the art is restricted from the perspective of its research scope. Except for PuLSaR, all other decision-making solutions [30, 77, 56, 55] are considered independent efforts apart from brokerage solution. [55] provides the application-specific (MapReduce) solution, which may not be effective on another kind of applications as well. Also, this solution does not target the comparison of infrastructure-level service of different cloud providers. PuLSaR is based on extended SMI method, which relies on customer specified details for ranking. Such methods can rank the cloud service irrespective of how the application will behave once deployed. Matrix is based on similarity matching of sampled traces. Sampling the performance data is an effective way to track the profile of application behavior. The basic strategy of Matrix is similar to the [64] and [29] as discussed in Section 2.3. However, cannot be considered time and cost effective considering the dimensionality of multi-cloud environment. A range of learning methods is applied in aforementioned state-of-the-arts. From the learning model perspective, sometimes the underlying relationship between the predictors and response variable is non-linear in nature and to derive such model require a lot of effort and expertise of this domain. Moreover, one learning model might not be considered effective for different applications or cloud providers and requires the need for multi-models based on different learning techniques.

### 2.4.4 Summary

In this section, we surveyed the work related to machine learning and its impact for decision support system. The research contribution discussed above are summarised in Table 2.2, which comprises of four columns. First column represents a reference to the state of the art. Second column indicates the objective of research work and proposed methodology. Third column presents machine learning methods utilised by the research, while the fourth column indicates the Quality of Service (QoS) attributes considered for decision-making using the machine learning methods. Similar to table 2.1 some additional dimension of summarised comparison is also added in this table. Last four columns indicate whether approaches are designed for single or multi-cloud environments; how much information has to be provided by the user; and how easy it is for each approach to respond to changes in the cloud market. Last column reports the application architecture for which the proposed approach is tested on. The purpose of this exercise was to ascertain that to what extent the state-of-the-art is answering the key questions raised in Section 1.5. The literature has shown evidence of the use of machine learning approaches in decision making as well as their effectiveness for predictions (Section 1.5-RQ1). Machine learning models have the ability to capture the data distribution for predicting the future behavior, which helps in observing application performance with different instances of input metrics (Section 1.5-RQ2). Some regularities are yet to be unfolded for unanswered questions, however, captured as requirements for designing a solution.

**Table 2.2:** Summary: machine learning aided decision support systems

| **State of The Art** | **Objective** | **Machine Learning Methods** | **QoS** | **Supported Cloud** | **Level of User Specified Info.** | **Response to Changes** | **Supported Apps.** |
|---|---|---|---|---|---|---|---|
| [72] | Dynamic Resource Scaling | RightScale Algorithms, Autoregression of Level 1 and Linear Regression | Availability and Cost | Multi | Performance metrics | May require parameter tuning and model tweaking | Network traffic |
| [73] | Resource Scaling | Pattern Matching Using KMP Algorithm | CPU Utilisation | 3 Grids | Specify threshold values | Model tweaking | Orchestrate video |
| [74] | Resource Provisioning | Linear Regression, Support Vector Machine, Neural Network for predicting workload pattern | Response Time, Throughput, CPU Utilisation | Single | Specify threshold in terms of response time and throughput | Model retraining based on new data | E-commerce benchmark |
| [75] | Auto Scaling | Linear Regression, Neural Network and sliding window to predict traffic surge | Workload, Traffic Surge | Single | None | Retrain model based on new data | Web server |
| [76] | Power Optimisation For Cloud Provider | Neural Network for power prediction | Energy efficiency | Single | Absolute power range | Parameter tuning and retraining | Data center |
| [77] | Resource Management For Cloud Provider | Support Vector Regression, Genetic Algorithm | Response Time | Single | SLA performance metrics | Model tuning and training | Not mentioned |
| Matrix [30] | Performance & Resource Management | Support Vector Machine For Clustering | Performance & Cost | Multi | Performance metrics | Model tweaking and retraining | CPU-intensive, memory-intensive, I/O intensive |
| Preference based cloud service recommender (PuLSaR) [27] | Decision Making Broker | SMI & Fuzzy Analytical Hierarchical Process | Qualitative & Quantitative | Multi | Fine-grained performance metrics | Model tweaking and retraining | CPU-intensive, memory-intensive |
| [56] | Cloud Service Provider Selection | K-Mean For Clustering | Pricing, Security, Virtual Machine Configuration | Multi | Coarse-grained metrics | Model retraining | Not mentioned |
| [55] | Resource Selection | Response Surface | MapReduce Performance | Single | Coarse-grained | May require model tweaking | MapReduce |

## 2.5 Discussion & Conclusion

In this chapter, we surveyed the background and related work aligned with the research goals stated in Chapter 1. The investigation was three-fold with a focus on a **multi-cloud broker**, **machine learning** and **decision support system** particularly for decisions regarding resource selection at the infrastructure level. First, the cloud brokerage offerings were investigated, looking at the state-of-the-art providing services at the different classified level of the broker. This provided the big picture of the broker services with ample examples to give the reader a wider level of understanding about the need for decision support system. Following this, methodologies of the decision support system were discussed to provide an overview of possible techniques to help cloud customer for decision making at infrastructure level in a cloud or multi-cloud environment. Lastly, the related work to the use of machine learning for decision making was investigated. To summarise, the following conclusion can be drawn from this chapter

1. The research efforts for decision support system are scattered around variable boundaries in cloud computing environment. Some of the efforts propose decision-making methods without considering the multi-cloud environment or any part of brokerage architecture. It is clearly depicted that very few efforts have targeted the decision support system as a brokerage service for multi-cloud environmental decisions. However, these solutions have lots of requirements to deal with. For example, users must have fine-grained information of their applications. They should be aware of all the business policies to show the impact of any change in the architectural design. Some decision support methods are based on ranking methods that ignores the fact of performance uncertainty. Moreover, these methods require the customer to specify application requirements in their own defined way like rating in numbers or assigning weights.

2. The decision support methods, which considers performance fluctuations using benchmarks or data traces cannot be considered cost and time effective keeping in view the dimensionality of cloud options. Though, such options could be very effective for making application-driven decisions, if aligned with learning strategies.

3. Machine learning can be applied for decision-making in a cloud environment in order to capture the application behavior on different deployment setups. However, there is a need to derive a generalised model or set of models, which can be applied to a range of applications. The design

of intelligent decision support system integrated with cloud-brokerage solution will be considered a productive effort to offer similar services as a package to the cloud-customer. Moreover, to design and develop learning based decision support solutions that are viable across the multi-cloud environment.

4. If one decides to apply machine learning methods on the collected traces, the traditional manner has to be followed which requires sufficient amount of data for testing and training purpose. If there is any change in the distribution of data the trained model is no more considered reliable about predicting the result. With the change in the distribution of data, the same traditional exercise has to be followed again starting from data collection, training, and testing which might not be considered an optimal way of applying learning methods. One should consider the learning approaches to transfer the knowledge between different learning scenarios. This would reduce the cost and time efforts.

Chapter 3

# Daleel, A Decision Support System

## 3.1 Overview

Chapter 2 has provided a holistic overview of the cloud brokerage, decision support systems and the use of machine learning for decision making. Unfortunately, existing approaches are lacking in many ways: **i)** user friendliness, **ii)** providing application-driven and realistic solutions, and **iii)** flexibility in terms of dealing with different cloud providers and application domains.

This chapter explores the design and architecture of Daleel, an intelligent decision support system, to specifically address the first research goal as stated in Chapter 1 and recalled here.

*"The designing of a cloud broker architecture integrated with an implementation of an intelligent decision support system."*

Firstly, a set of important design requirements are constructed by observing the limitations and challenges of existing approaches. Secondly, an abstract overview of the proposed solution is given based on these requirements. Thirdly, the design principles at the core of Daleel are discussed. This includes the construction of machine learning aided decision support module and its methodological steps to achieve the learning models that are viable across different application domains.

Now, recall second research goal, as stated in Chapter 1:

*"The investigation of machine learning methods that can be applied for optimal decision making in the decision support system of a cloud broker."*

To address this research goal, a machine learning aided decision support module is constructed and its methodological steps are explained in order to build learning models that are viable across different application domains. Lastly, the representative machine learning methods are discussed to answer key research question of second goal, regarding applicability of learning methods for designing a decision support system in a cloud environment.

## 3.2 Problem Formulation

The decision support system should be an integral part of any cloud brokerage framework. The decision support module should have a capability to assist the customer by providing realistic as well as application-specific decisions related to cloud and service selection. This module should provide valuable behavioural insight into the application performance necessary for making optimal decisions. Moreover, the architecture of decision support module should have the capacity for large-scale analysis related to any decision making process. The decision-making process should be presented to a customer in an easy way by abstracting details of any computational complex methods. The user should not be asked to provide any fine-grained application specific details or configuration requirements.

## 3.3 Proposed Solution

Daleel is a solution devised to provide a decision support system to assist customer for optimal deployment decisions in a way that meets the requirements set above. The proposed solution is devised to meet the above-mentioned set of requirements and based on the following design principles.

### 3.3.1 Overview of Daleel

Daleel offers a decision support system integrated with the cloud brokerage. It is specifically designed to provide assistance to the cloud customers for optimal decision making related to application deployment. More precisely, decisions related to the selection of optimal IaaS provider and instances.

The main design principles of Daleel are twofold. The first principle is to equip customer with some evidence-based knowledge considering the black-box nature of cloud offerings. As, cloud providers do not provide any necessary details such as scheduling algorithms, the parallel workload on virtual machines or how virtual cores are pinned to physical cores. Therefore, machine learning provides necessary insight for decision making. The second design principle of Daleel is to abstract the complexity of the decision making steps from customers. The solution requires the high-level specification of an application from the customer, collected as application vignette. This information is collected in a user-friendly and understandable format. The application vignette (general description) is a short set of key-value pairs provided by the customer that serves as a high-level description of the application requirements. This also includes high-level descriptors of the application to categorise it as either memory-intensive, CPU-intensive, memory-CPU intensive and so on. The customer related constraints are also collected regarding the QoS attributes. The customer constraints include the customer's functional and non-functional requirements, such as minimum QoS, availability, location, and budget. A cloud provider's portfolio contains data that we obtain (through APIs and web scraping) based on their resource provisioning levels, resource meta-data, and pricing models.

## 3.3.2 Architecture of Daleel

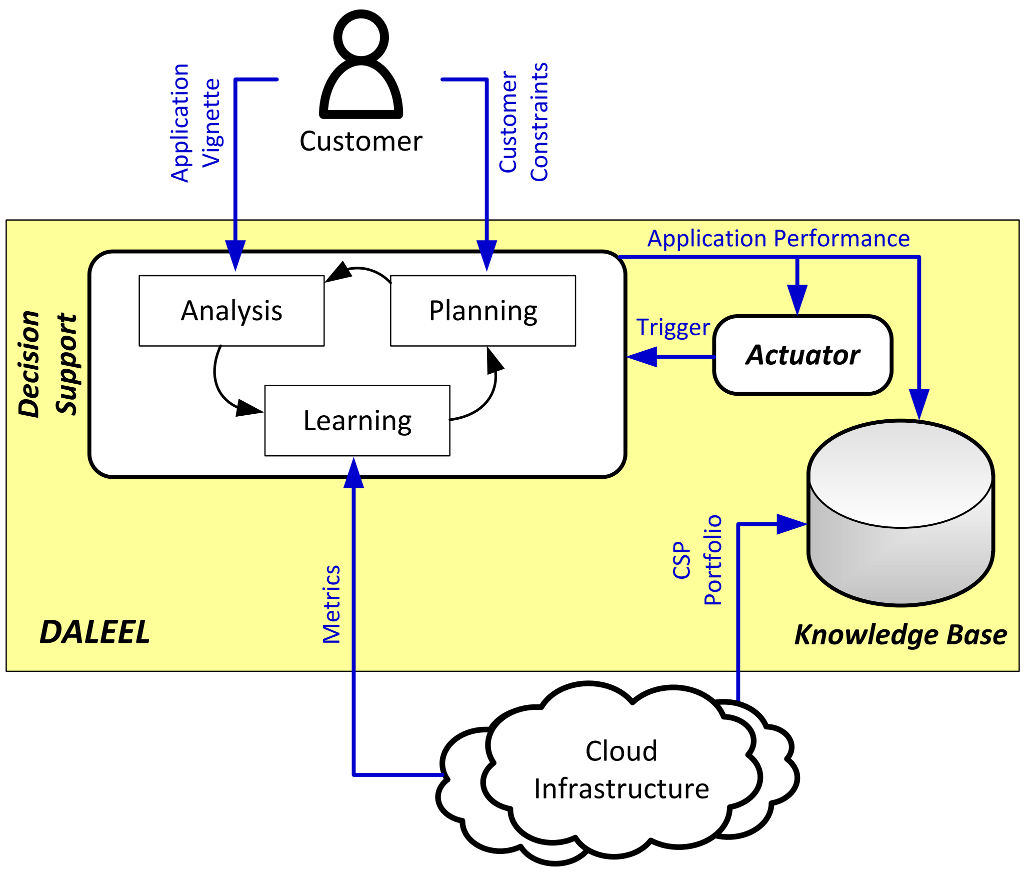


**Figure 3.1:** Daleel's architecture.

The basic architecture of Daleel, depicted in Figure 3.1, consists of three primary architectural elements: *Decision Support*, *Actuator*, and *Knowledge Base*. The **Decision Support** module, equipped with machine learning models, is at the heart of Daleel's architecture. This module relies on a three-phase process that continuously operates throughout the application life cycle to predict application performance. These phases are: *Analysis*, *Learning*, and *Planning*. Each phase carries out different yet complementary operations to acquire deep knowledge of the available cloud deployment options and how suitable they are for a given application. The detailed functioning and architecture of these three phases is revealed in next section. Daleel supports incremental and iterative design feature for development of application specific as well as generalised learning models based on the analysis of different applications deployed on cloud IaaS. The generated learning models are then used for predicting application performance to assist the user in making application-driven decisions. The **Actuator** triggers the Decision Support module into operation at different times. This could be based on thresholds set according to the customer constraints on application QoS, application load, or Knowledge Base information (change in a provider's portfolio). Such triggers will launch new Analysis and Learning cycles, or will activate the Planning logic to begin migration to a new cloud infrastructure. Migration between different cloud infrastructures is a big challenge in its own right and is outside the boundaries of this work. However, the Planning logic could easily be extended to incorporate migration methods [78]. The **Knowledge Base** holds data collected by the Decision Support module. This data is comprised of profiling traces, cloud portfolios, an application vignette, learning models and learning settings/ parameter settings.

## 3.4 Daleel's Decision Support

The three phases of Decision Support module are responsible to perform important tasks in order to provide an optimal deployment decision. A detailed modular description of these phases are described in Figure 3.2.

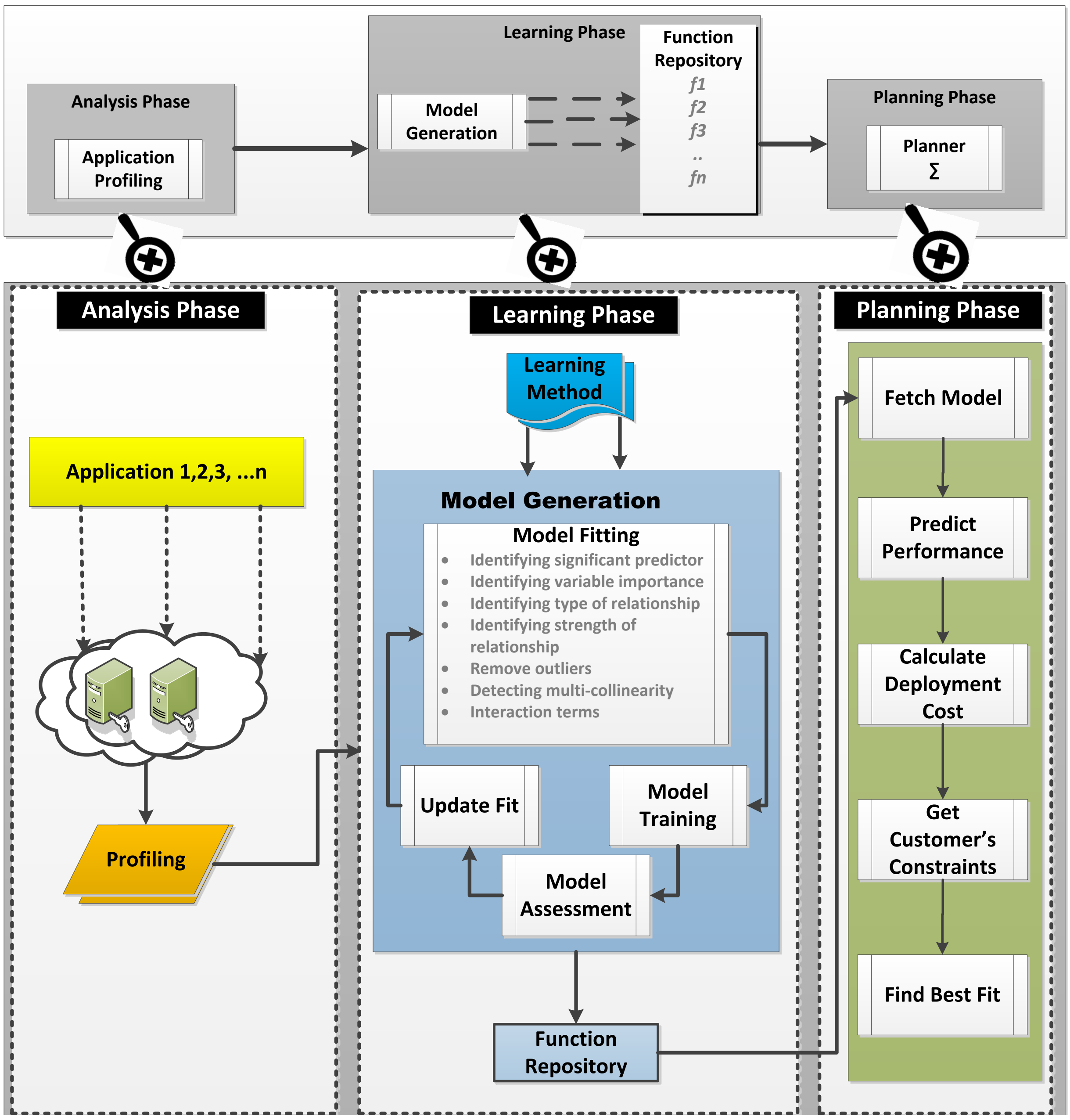


**Figure 3.2:** Overview of the three phase decision support system.

### 3.4.1 Analysis Phase

The Analysis Phase kicks off the process of decision-making. This stage contains the *Application Profiling* module that records the time-series traces of the application's and the infrastructure's performance. Data profiling is an effective way of tracking the application's behaviour under different deployment setups. This can be carried out *live* on a shared cloud infrastructures (whether public or private), or *offline* in a completely controlled and isolated virtual environment.

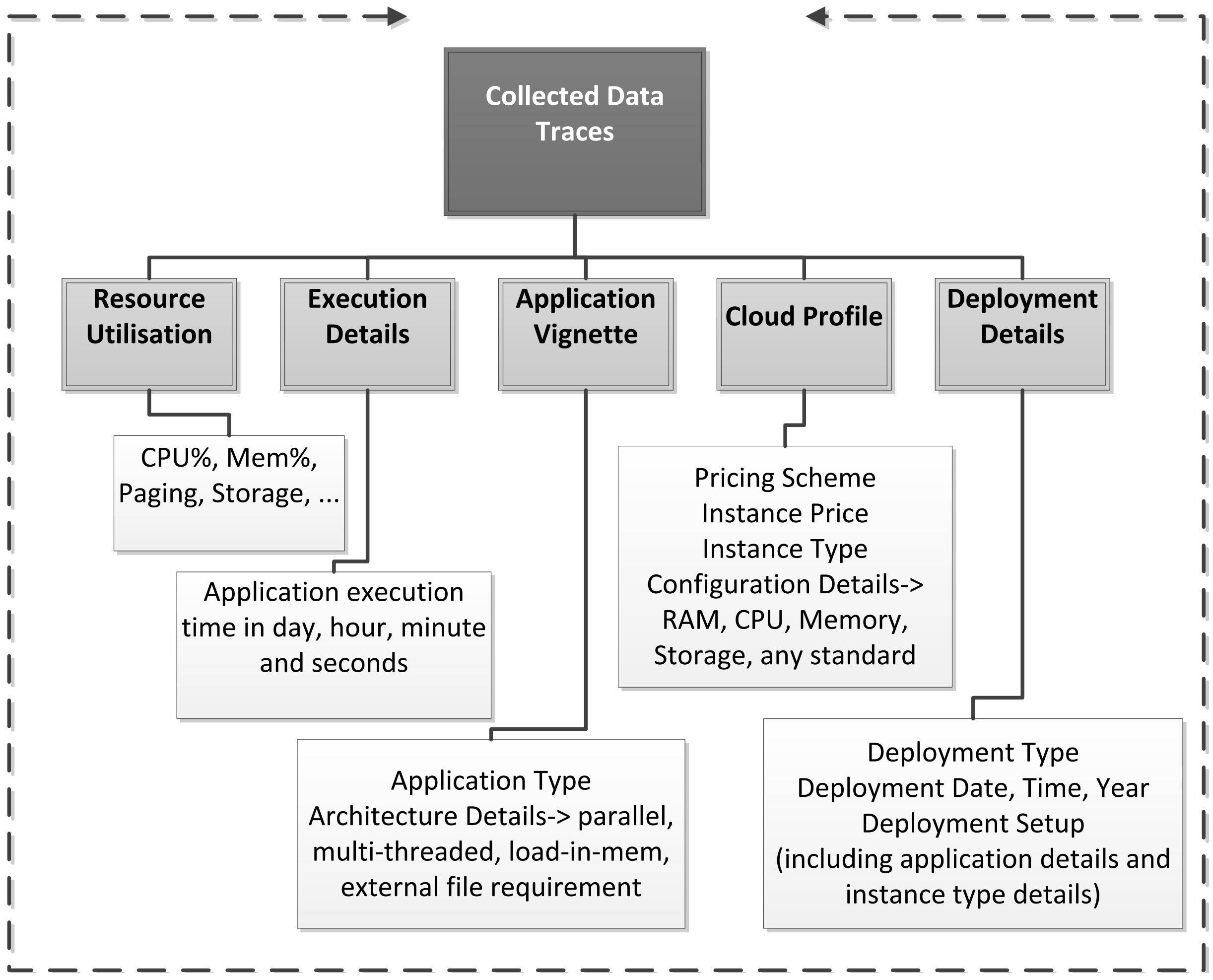

Collected Data Traces
Resource Utilisation
Execution Details
Application Vignette
Cloud Profile
Deployment Details
CPU%, Mem%, Paging, Storage, ...
Application execution time in day, hour, minute and seconds
Application Type Architecture Details-> parallel, multi-threaded, load-in-mem, external file requirement
Pricing Scheme Instance Price Instance Type Configuration Details-> RAM, CPU, Memory, Storage, any standard
Deployment Type Deployment Date, Time, Year Deployment Setup (including application details and instance type details)

**Figure 3.3:** Collected data Traces details

The profiling data contain information about the application's performance on a representative set of selected virtual machines. Each dataset is extracted as an outcome of individual experiments targeting the specific application's deployment on a subset of selected virtual machines (EC2). The collected data traces are composed of information about the application vignette, cloud portfolio and details regarding deployment, execution and resource utilisation, as shown in Figure 3.3. Application vignette sustains information about the application type such as whether the application is memory intensive, CPU intensive or a mix of both. It also holds information about the application's architecture such as whether it is single threaded, parallel or multi-threaded. Other details that exist include the external file requirement, external file size and in-memory computation. Deployment details include information about the deployment date, time, day and year. Execution details include application execution time in days, hours, minutes and seconds. Utilisation details include CPU utilisation, memory utilisation, paging, storage,and so on. Cloud profile include information about instance specification and pricing scheme. Aggregating different application profiles builds up the Knowledge Base with information about the application's description and its behaviour on different deployment setups.

**Application profiling** module is responsible for collecting profiling information by deploying the application on different virtual machines. Following activities are involved in this process:

1. **Application vignette -*getAppVignette()*:** A user has to provide application executable files along with its vignette. *getAppVignette()* function collects following application-specific information from user and writes to a CSV file. <*int applicationId*>: id is automatically assigned to each application. <*String applicationName*>: name of the application has to be provided by the user. <*String applicationType*>: the user has to choose the type of application from a displayed list such as IO, CPU, Memory, etc. In this research, we have used three representative applications of three types, one is CPU-intensive, second is memory-intensive and third is a mix of CPU and memory. <*boolean parallel*>: the user has to specify if the application architecture exploits parallel architecture. <*boolean multiThreading*>: this information keeps track if the application is single-threaded or multi-threaded. <*boolean loadinMemory*>: the user has to identify if the application execution requires some file to be loaded into memory, as the case in memory-intensive applications. <*boolean externalFilerequired*>: it has to be specified if the application requires some input file to be used during application execution. <*int fileSize*>: if the application requires some supporting file (as input) what is the size of the file.

2. **Create VM - *createVM()* :** A VM can be launched through a portal or *createVM()* function which makes use of provider Java SDK. In this research, the representative cloud providers are Amazon and Google. To create a virtual machine following parameter values need to be provided: <Credentials>: it includes the <access-key> and <secret-key>. <*setInstanceType(String instance)*>: every provider offers different categories of virtual machines specific to different application needs, for example, t2.small is an instance type in Amazon EC2. In this research we chose a representative set of virtual machines from different categories, the details can be found in Section 4.3.2. <*setMinCount(int mincount)*>, <*setMaxCount(int maxcount)*>: this number explains the number of virtual machines to be created for each type. <*setAvailabilityZone(String*

*zone)>*: A user can choose any availability zone. The selected availability zone in this research is *EU-west*. *<setImageId(String imageid)>*: this is operating system image a virtual machine would have when launched. In this research, we used 64 bit Ubuntu Linux14.04. *<setKeyName(String keyname)>*: name of the secret key has to be provided by the user.

3. **Deploy application - *deploy.sh*:** a bash script is written to deploy the application. This script provides support to deploy executable of a standalone application. Deployer has to make sure that all the executables, scripts and supporting files are in the same location to be copied to VM through SCP. This script can be included during VM creation process through a portal or can be executed through terminal once the VM is created. This script is also responsible for monitoring application elapsed time in each iteration and logs output in a text file (applicationlog.txt) which is stored in the local storage of VM. The application executes for a given number of runs and a delay can be added using a *sleep()* function in each pair of the run. Further details about number of runs and delay are given in Chapter 4. Following are two example commands in the script to log application execution date and elapsed time.

   ```
   $ date | tee −a applicationlog.txt
   $ time −o applicationlog.txt −a java -jar applicationlnx.jar
   ```

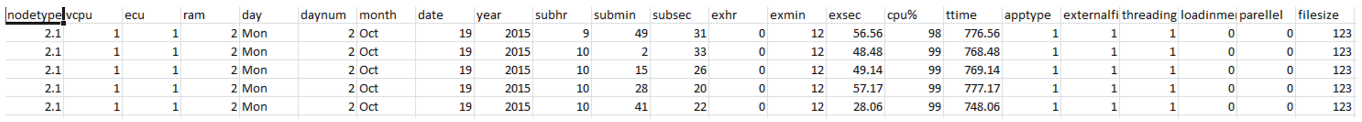
4. **System monitoring - *monitor.sh*:** System monitoring script logs information about the virtual machine specifications, few script snippets are shown below.

   ```
   $ echo "*****CPU INFO********" |tee -a cpuout.txt
   $ cat /proc/cpuinfo |tee -a cpuout.txt
   $ echo "*****MEMORY INFO********" |tee -a cpuout.txt
   $ cat /proc/meminfo |tee -a cpuout.txt
   $ echo "*****CPU HARDWARE********" |tee -a cpuout.txt
   $ lscpu |tee -a cpuout.txt
   ```

   **vmstat** and **sysstat** are two system monitoring tools used to monitor the system performance during the time of application execution on a virtual machine. *vmstat* traces are logged in a text file (vmstatlog.txt) after every 3 seconds. On the other hand, *sysstat* keeps track of system traces after every minute for 24 hours which can be saved in a syststoutput.log file. *sysstat* is a system performance tool for Linux. It includes several system performance tools like iostat, sar, mpstat, pidstat, sadf. **Sar** collects system activity information, **iostat** informs CPU utilisation and disk

I/O statistics, *mpstat* reports pre-processor statistics, *pidstat* informs about Linux processes and *sadf* is responsible to display stats collected by sar. The stats collected by sar informs about I/O transfer rate , paging activity, interrupts, network activity, memory utilisation, CPU utilisation and kernel activities. All the monitoring data is extracted through a SCP command and saved on a specified private storage.

5. **Parsing log files:** The above mentioned collected data is in raw format and requires transformation to a comma delimiter CSV file to be used by the Learning module. The *parseApplication(path-to-.txt-file)* and *parseVmstat(path-to-.txt-file)* functions (written in Java) read the txt files and write the extracted output into a .CSV file along with application vignette. The resulting CSV file contains information about deployment details, cloud profile, execution detail and application vignette, as shown in Figure 3.4 with some selected columns to be displayed. Some of the factor based values, for example, application specific parameters, are transformed into numerical values for the sake of reducing data complexity for model generation.



| nodetype | vcpu | ecu | ram | day | daynum | month | date | year | subhr | submin | subsec | exhr | exmin | exsec | cpu% | ttime | apptype | externalfi | threading | loadinme | parellel | filesize |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2.1 | 1 | 1 | 2 | Mon | 2 | Oct | 19 | 2015 | 9 | 49 | 31 | 0 | 12 | 56.56 | 98 | 776.56 | 1 | 1 | 1 | 0 | 0 | 123 |
| 2.1 | 1 | 1 | 2 | Mon | 2 | Oct | 19 | 2015 | 10 | 2 | 33 | 0 | 12 | 48.48 | 99 | 768.48 | 1 | 1 | 1 | 0 | 0 | 123 |
| 2.1 | 1 | 1 | 2 | Mon | 2 | Oct | 19 | 2015 | 10 | 15 | 26 | 0 | 12 | 49.14 | 99 | 769.14 | 1 | 1 | 1 | 0 | 0 | 123 |
| 2.1 | 1 | 1 | 2 | Mon | 2 | Oct | 19 | 2015 | 10 | 28 | 20 | 0 | 12 | 57.17 | 99 | 777.17 | 1 | 1 | 1 | 0 | 0 | 123 |
| 2.1 | 1 | 1 | 2 | Mon | 2 | Oct | 19 | 2015 | 10 | 41 | 22 | 0 | 12 | 28.06 | 99 | 748.06 | 1 | 1 | 1 | 0 | 0 | 123 |

**Figure 3.4:** Transformed values in a CSV file.

### 3.4.2 Learning Phase

The second phase in the Decision Support module is the Learning phase, which is comprised of the two elements shown in Figure 3.2; the *Model Generation* element and the *Function Repository* element. The Learning Phase aims to generate a prediction function to accurately predict the application execution time which can further be used to calculate the cost of application execution. It also aims to calculate the correlation between predictors and responses so as to infer a relationship between the two and use this relationship for future predictions.

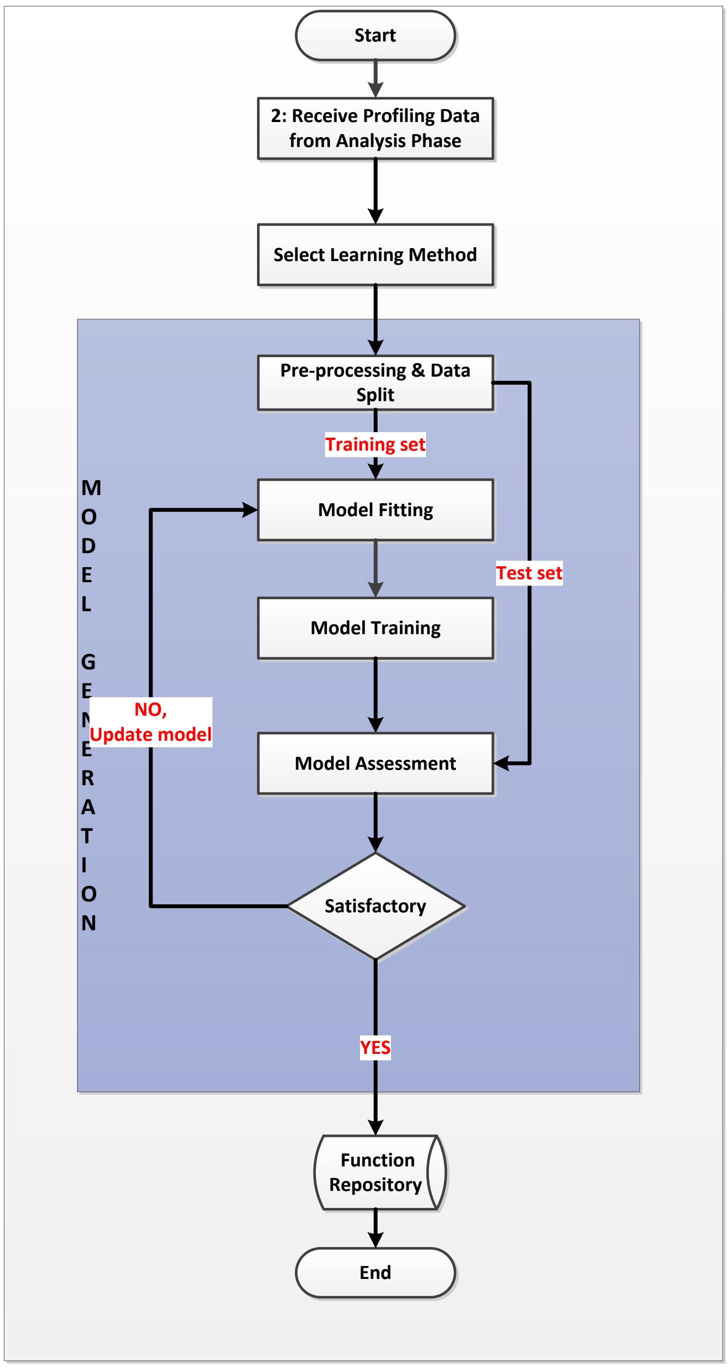


**Figure 3.5:** Flow chart of the Learning phase.

Figure 3.5 explains the flow of steps to generate a learning model to predict application execution time. Learning phase starts by receiving data (.CSV file) form Analysis phase and selects a machine learning method to start development process of a learning model. The third step is a selection of a machine learning method. In this research different machine learning methods are used to generate prediction and inference functions for multiple applications such as $f1, f2, ..., fn$, and are stored in the Function Repository. The machine learning methods used in the Learning phase are discussed in Section 3.5. The Model Generation module starts with pre-processing of data to remove any empty or wrong values within the .CSV file. Pre-processed data is then split into training and test set. The *Model Generation* module comprises a set of processes that are executed in an iterative way, as shown in Figure. These processes are *Model Fitting*, *Model Training*, *Model Assessment* and *Update Fit*. Amongst all of these, *Model Fitting* is the most important and time-consuming phase of generating a model that is explained in detail ahead. Model fitting process received training data and generate a learning model. The fitted model is then get trained and passed on to the model assessment which evaluates the trained model on test data set. If the prediction results are satisfactory the model is saved into function repository and in case of unsatisfactory results, model starts update phase.

Model generation follows the traditional principle of learning a model that includes the presence of data traces (generated by Analysis phase) which belong to the same data distribution set. This means the decisions are related to one type of application for a particular cloud provider.

1. **Model Fitting**

   Model Fitting is comprised of following steps as shown in Figure 3.2. Algorithm for Model Generation module is stated in Chapter 4 (Algorithm 1) and details pertaining to each of these steps are described here. An R markdown script explaining R code with respect to each of the stage is presented in Appendix B. All steps can be performed automatically through the markdown script, however, a human intervention is required at few places to assess results. Assessment criteria is mentioned at each stage to be understandable by a human having less or no expertise in machine learning.

   (a) **Identifying significant predictors**.

   *Q: are any of the predictors X useful in predicting the response Y?, and which of the predictors are associated with the response?*

   This can be achieved by hypothesising the relationship R and to check for the contribution of each predictor in that relationship. A relationship R can take any mathematical form to describe the relationship between dependent variable $Y$ and independent variable(s) $X$.

For example, one can start with a hypothesis $H_i$, indicating a significant linear relationship $R$ between the independent variable $X$ and the dependent variable $Y$. In contrast, the null hypothesis $H_0$ states that there is no relationship between the predictors and the response variables. The linear relationship $R$ can be described mathematically

$$Y \approx \beta_0 + \beta_1 X \tag{3.1}$$

"$\approx$" can be read as "approximately modelled as", in other words, one can say that Y is regressing on X. In equation 3.1, the terms $\beta_0$ and $\beta_1$ represents model *coefficients* (constant values) and indicate the *intercept* and *slope* terms in a linear model.

We test the null hypothesis,

$$H_0 : \beta_1 = \beta_2 = ... = \beta_p = 0 \tag{3.2}$$

and the alternate hypothesis

$$H_1 : \text{ atleast one } \beta_j \text{ has a non zero value} \tag{3.3}$$

Some fundamental statistical methods are used to evaluate results of hypothesis, for example, F-statistics, p-value, Residual Sum of Square (RSS) and Total Sum of Squares (TSS). Details for each of the method is explained in Appendix A.

**lm() function:** a linear function in R can be generated using this built-in function that tries to create a linear relationship between response and predictor variables, as shown below.

$> Model = lm(< responseVariable > < predictors >, data = trainingData)$

Summary of the generated function can be displayed using *summary()* function that takes a model as an input.

$> summary(Model)$

The summary can display all the learned coefficient values, F-statistics, p-value ,and R-value for the model. The actual implementation of this function with real values along with interpretation instructions is detailed in Appendix B.

**glm() function:** a non-linear function in R can be generated using this built-in function that tries to create a non-linear relationship between response and predictor variables with different polynomial order, as shown below.

$> Model = glm(< responseVariable > < poly(predictors, degree) >, data = trainingData)$

(b) **Verifying variable importance**.

*Q: Do all the predictors help to explain Y, or just a subset of predictors are considered significant?*

Variable selection determines that which of the predictors are associated with response to fit a model and are significant for deriving a robust model. Variable selection could be done by comparing a lot of models, each containing a different predictor subset. Forward selection and backward elimination of variables are used to select a subset of predictors that are considered important to define a relationship between predictors and response variables. The P-value is one of the key measures used to determine the importance of the variables that contribute towards defining a strong relationship. The P-value associated with each predictor is evaluated and whichever has a lower value is selected for the model generation. Ideally, variable selection is performed by trying various models that contain different subsets of predictors. The $R^2$-value, which is also known as the coefficient of determination, is a statistical method for measuring the closeness of the actual and predicted data in terms of how similar the actual data are to the model fitted line or curve.

**regsubsets() function:** this built-in function can automatically apply forward and backward selection method to identify important predictors. It takes a dataset as an input with the identification of response variable and selected method (forward or backward).

$> ForwardSelection = regsubsets(< responseVariable > ., data = trainingData, nvmax = 5, method = "forward")$

$> summary(ForwardSelection)$

$> BackwardSelection = regsubsets(< responseVariable > ., data = trainingData, nvmax = 5, method = "backward")$

$> summary(BackwardSelection)$

**regsubsets() function:** can also be used to identify subsets of predictors showing best relationship with the response variable. In the example, it tries to find 7 best subsets comprising different combinations of predictors. The implementation of this function with real values can be found on page 2 of Appendix B.

$> nbest = regsubsets(< responseVariable > ., data = trainingData, nbest = 7)$

$> summary(nBest)$

The list of significant predictors as a result of this research are shown in Chapter 4, Table 4.3.

(c) **Identifying type of relationship**.

*Q: Is the relationship always linear?*

It is not necessary that the true relationship between the responses and predictors is always linear. Residual plots are used to identify if there is a chance of non- linearity in the relationship.

**residualPlots() function:** residual plots are graphical representations of residuals on the vertical axis and the independent variable(s)/predictors on the horizontal axis.

$> residualPlots(Model,\ 1, fitted = TRUE)$

The term residual indicates the difference between the observed value of the response variable(y) and the predicted value (y). Residuals are one of the bases for most of the diagnostic methods. For example, if we have hypothesised a linear relationship between the response and predictors and if the hypothesis is true then the points in the residual plots are randomly spread around the horizontal line to indicate a linear relationship. On the other hand, if the true relationship is not linear, patterns or curves are observed in the plot, thus indicating non-linearity. This plot is very useful in exploring the combined and individual relationship of response(s) and predictor(s). The plot shows no pattern if the relationship is linear.

**marginalModelPlots() function:** Component- Plus-Residual (CPR) or marginalModel plots are considered to be an effective technique to find non-linear relationships between individual predictor(s) with response variables.

$> marginalModelPlots(Model)$

If the relationship denies the chance of linearity then other functions can be used to summarise the association of response(s) and predictor(s). This must follow the same steps starting from the hypothesis generation.

**avPlots(), crPlots() function:** Added variable graphs are good to see the effect of each regressor after adjusting for all other regressors and shows the impact of observations on regression coefficient.

$> avPlots(Model, id.n = 2, id.cex = 0.6)$

$> crPlots(Model)$

crPlots function tries to fit smoothness function (polynomial order) on each of the predictor to identify the range of non-linearity.

Further explanation of these functions (R-built-in functions) can be seen on page 15 onwards of Appendix B. This phase requires a human support to manually scan the generated plotted results.

(d) **Identifying strength of relationship**.

*Q: How well does the model fit the data?*

The strength of the relationship is measured using the model's accuracy. Two measures are used to check the model accuracy, one is by using the Residual Standard Error (RSE) and other is by using $R^2$. Above two stages can identify if the relationship is linear or non-linear. Further to that non-linearity can be checked by transforming predictors as well as applying different non-linear functions, for instance, a polynomial of a different order.

**glm()** function can be used to check the model fitness with inclusion of any polynomial order. More explanation of this type of function can be found in Appendix B.

$> Model = lm(< responseVariable > \ poly(predictorVariable, 3)+poly(predictorVariable, 2)+ ..., data = trainingData)$

**svm() function:** This built-in function provides support for regression as well as classification problem. This function has capability of operating with different kernel methods, such as, Radial, Polynomial, etc. This function can take any number of parameters with kernel function and tries to fit a model by mapping a function to high dimensional hyperplane. Following example code tries to learn a function with polynomial of degree three with certain parameters that help to tune a model. Details of these parameters can be found in Section 4.3.3.

$> svm(< responseVariable >< predictors >, data = trainingData, scale = FALSE, kernel = "polynomial", degree = 3, gamma = x, cost = 0.1, coeff = 1...)$

The generated learning models for the representative applications using glm() and svm() functions are listed in Section 4.3.1 and 4.3.3.

Regularization methods can be used to prevent overfitting of a function and can improve generalisability of the learning model. One can apply most common forms of regularization methods, such as Ridge and Lasso. Further details can be found in Appendix B (page 11) and in [39] where we applied these methods on the generated learning models.

Response and predictor transformation is a way of transforming the non-normal distribution of dependent variables into normal. Normality is an important assumption before applying large set of statistical test methods. One can use Box-Cox transformation for response variables and Box-Tidwell for predictor transformation as detailed on page 41 of Appendix B.

(e) **Remove Outliers**.

*Q: Is it necessary to remove outliers? Are they leading the relationship towards negative results?*

An outlier is an observation that is far away from the random variables in the population sample. This could be due to some variability in the dataset or some measurement error. Outliers should be investigated carefully as sometimes they contain useful information about the dataset. Box plots and some statistical methods are used to identify any outliers in the dataset. The removal of outliers is sometimes necessary as they can influence the coefficient measurement. Outliers are identified in the box plot using the interquartile range and greatly effect the mean of the dataset. They also show a non-normal distribution with a heavy or long tail that is indicated in the distribution plot. Outliers can either be truncated or replaced with the nearest possible value that can be accommodated in the dataset while not influencing any relationship.

**spreadLevelPlot() function:** Outliers can be identified using *studentized residual*, which is a form of t-statistics and is the quotient resulting from the division of a residual by an estimate of its standard deviation. This function takes a model as an input, as shown below.

$> spreadLevelPlot(Model)$

For this research, we have removed all the data set values having more than double of standard deviation in the response time.

(f) **Detecting multi-collinearity**.

*Q: What is the impact of multi-collinearity on the relationship?*

Multi-collinearity or collinearity refers to the high correlation of two or more predictor variables in a regression model setting. For example, when one of the predictor variable can be linearly predicted from the other. Multi-collinearity does not affect the predicting power of a model as a whole although it affects the coefficient values regarding an individual predictor. This can result in a large amount of standard errors in the model.

**vif() function**: collinearity in regression analysis can be detected using the Variance Inflation Factor (VIF) which measures the increase in variance of estimated regression coefficients due to collinearity. Statistical details can be found in Appendix A and an example use of this function is shown below.

$> vif(Model)$

The result of this method is a collinearity factor, higher the value the more chance of collinearity. This inflation can be tackled by removing predictor of higher inflation factor.

(g) **Interaction terms**.

*Q: Can interaction terms be used to derive a relationship?*

A synergy exists between different sets of predictors that contribute to defining a relationship with a response. A standard linear model assumes an additive relationship between response(s) and predictor(s). Such models clearly indicate the effect of the predictor on the response and are easy to interpret in terms of the relationship. The use of interaction terms in the model sometimes increases the R2 value; in such a case, interaction terms are considered a substantial addition to the model. The generated learning models with interaction terms are listed in Section 4.3.1 and details are listed on page 50 of Appendix B.

2. **Model Training**

Model training involves training the fitted model function $f'_{Train}$ on training dataset with fixed parameters that are calculated during model fitting stage. We train and assess the accuracy of the model through different testing measures: p-value, $R^2$, Residual Standard Error (RSE), and F-statistics. R2 measures the proportion of variability in the response variable that can be explained by the predictors. The RSE shows the actual deviation of the response from the predicted value and measures the lack of fit for the model [79]. Although the $R^2$ value shows the goodness of fit, it cannot assess how accurately the predictors can estimate the response. Therefore, resampling methods such as cross validation and bootstrapping are employed on the training set for model assessment and model training [80]. Resampling methods repeatedly draw samples from the training set and refit the model on each sample to get additional information about the fitted model's performance, such as variability estimates of regression fit [81]. Cross validation is one of the widely used resampling methods for model selection. We used the k-fold cross validation method, computed by averaging the Mean Squared Error (MSE) for k-folds over the test sample. Statistical details of cross validation method and MSE can be found in Appendix A.

**ModelTraining(trainingData, Model) function:**

a code snippet of this function is shown below. the sample code will train a model in 200 iterations where each of the iterations will have a 10-fold cross-validation of data for 200 times.

$> for(1:200)$

$> MSEtraining = cv.glm(trainingData, Model, k = cvValue)\$delta[1]$

3. **Model Accuracy**

   Model accuracy is evaluated on the test data set that is not involved in any fitting or training process.

   **ModelTesting(testData, Model)**: the test set is considered a validation set in order to assess the strength and relationship of a predictive function that is derived using the training dataset.

   Following code calculates the mean squared value of actual and predicted data on the test dataset.

   $> MSEtest = mean((responseVariable.testData - predict(Model, testData))^2)$

   If the testMSE is less than the trainingMSE value the model is accepted, this also indicates that the model is not over-fitted.

4. **Update Fit**

   Update fit is the process of repeating the model generation and specifically tuning the model parameters in order to refine the results. This stage influences the next iteration using the evaluation results of the previous iteration. This phase is responsible for two things: for starting a new iteration of the model generation step with a different dataset or machine learning method and for ending with agreeable model results.

### 3.4.3 Planning Phase

The third phase in Decision Support module is *Planning Phase* as shown in Figure 3.2. This phase takes input from the Learning phase in the form of a prediction model which generates a vector output based on the input requirements of the customer. The *Planning phase* is designed to support a multi-criteria decision making problem, where a set of vectors describing the performance is the learning outcome. Multi-criteria resolution does not come under the scope of this work and this thesis targets two QoS attributes as our criteria, namely VM price and application performance. Therefore, the planning logic takes into account these two QoS attributes and provides recommendations for the application's deployment. The planning process outputs the deployment options following these steps as shown in the right hand box in Figure 3.2:

1. Fetch the learning model $f$ from *Function Repository.*

2. Predict application performance on a set of inputs. predict() function takes two inputs, one is a prediction model, and second is a data frame comprised of predictor variables which are part of the prediction model.

   $> new.data < -df.frame(predictors)$

   $> predict(Model, new.data)$

3. Calculate the price for each deployment option according to the cloud metrics.

4. Get customer constraints. In this research, we are informing customer with a list of output showing price and performance for each of the nodes in incremental order.

5. Find the best fit criteria according to constraint.

## 3.5 Machine Learning Methods for Model Generation

This thesis focuses on the use of supervised machine learning technique where the known training dataset is used for generating a prediction function which then is used for predicting future values. The proposed architecture of Daleel supports the supervised learning methodology where the datasets are provided by the *Analysis Phase* in the Decision Support module. The motive for using machine learning methods in the Decision Support module is three-fold: **i)** provision of evidence-based knowledge, **ii)** capture of realistic performance behaviour and **iii)** understanding of behavioural relationships. Considering these motives different machine learning algorithms are explored.

In context of this research, statistical inference methods are explored to understand the relationship of application with the deployment characteristics: for example, to understand the relationship of application performance with respect to change in the virtual machine configurations. This information is valuable and provides necessary insight for realistic decision making. An inference method estimates how Y changes as a function of X. In this case the estimated function is not treated as a black-box because it presents all the necessary relationship details.

The statistical inference is used to answer either questions such as: **i)** which predictors are associated with response? **ii)** what is the relationship of response and predictor? **iii)** what is the strength and type of relationship? and so on. The inference methods follow a statistical approach which starts with an assumption about the relationship of response variable and predictor and tries to fit the model. To some extent, this process is time consuming as it requires initial assessment about assumptions and an inevitable repeating of the model fitting process multiple times. This process provides valuable information to capture behavioural change in application performance as required for optimal decision making. An additional advantage of applying inference methods is reduction in the feature space by identifying the significant independent variables. The process of inference estimates with some initial assumption is categorised as a parametric method [80]. In contrast, non-parametric methods do not make explicit assumptions about the relationship of X and Y. Instead they try to estimate a function that gets as close to the data points as possible. Non-parametric methods choose flexible models that can fit many different possible functional forms to estimate the function. In general, fitting a more flexible model requires estimating a large number of parameters which makes the model more complex. One advantage of non-parametric methods is the flexibility to estimate any given function. This however, requires a large number of observations for function estimates.

Considering both advantages and disadvantages of parametric and non-parametric methods, this research tries to make use of both model generation methods for supervised machine learning. The architecture of the proposed decision support system supports both inference and flexible methods. **Multiple Polynomial Regression (MPR)** is used to get necessary insight about the relationship and to estimate an inference function using the least square function.On the other hand, **Support Vector Machine (SVM)** in regression setting is used to take advantage of flexible methods. In addition, SVM are used due to two main advantages: i) it has a regulaization parameter to avoid over-fitting, and ii) it uses the kernel engineering to generate expert knowledge.

This sections presents fundamental details about the selected machine learning methods that are used for model generation in the decision support system of Daleel.

### 3.5.1 Multiple Polynomial Regression

Regression analysis traces the distribution of a response variable ($Y$) (or some characteristics of this distribution, such as the mean) as a function of one or more explanatory variables ($X_1 ... X_n$). The response variable is often referred to as the dependent variable and the explanatory variable is referred to as the predictor or independent variable. Regression analysis explores the relationship of response variables with the explanatory or predictor variables.

The predictors can be numerical variables, such as: height or age on an in- formation sheet; qualitative variables, such as sex, country of origin or application category; ordinal variables, such as assessment scale on a range of 5 points. In regression analysis, the predictors are converted to regressor variables, which are numerical in nature. For example, a qualitative variable with $n$ distinct level requires $n-1$ regressors. However, a numeric predictor corresponds to one regresssor that is the predictor itself. In some cases, regressors may require transformation to another scale using logarithms or polynomial equations.

**Multiple regression** extends the concept of simple linear regression by allowing more than one predictor or regressor in a linear regression model. This can be done by giving a separate slope coefficient to each predictor. The multiple linear regression can be represented mathematically as shown in Eq 3.4:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon \tag{3.4}$$

**Polynomial regression** is the extended form of a linear relationship to accommodate a non-linear relationship between response and predictor. Such a model for a single predictor $X$ is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X^2 + ... + \beta_h X^h + \epsilon \tag{3.5}$$

Here $h$ refers to the degree of the polynomial in one variable equation.

A quadratic regression equation can take up the following mathematical form as given in Eq 3.6

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X^2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 + \epsilon \tag{3.6}$$

Suppose there are 50 data points in a data set and the relationship among the data points is explained using quadratic regression, the matrices for the second order degree polynomial model can be described as shown in Eq 3.7 and 3.8:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{50} \end{pmatrix} \qquad X = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ \vdots & \vdots & \vdots \\ 1 & x_{50} & x_{50}^2 \end{pmatrix} \tag{3.7}$$

$$\epsilon' = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_{50} \end{pmatrix} \qquad \beta' = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_3 \end{pmatrix} \tag{3.8}$$

Polynomial regression is an important form of non-linear regression where the relationship between response and predictor can be modelled by a quadratic function. A polynomial of order K can have k-1 bends is the regression line. Most often, polynomial of degree less than 4 are employed for data analysis and model generation.

If the values of qualitative variables are categorical labels rather than measurements or numeric values, we refer to these qualitative variables as factors and the categories as levels. While generating a regression/linear model, the factors may be be used by transforming them into dummy values. These dummy values are numerical and their range is equal to the number of categorical levels. For example, in a clinical environment that dispenses medicinal tablets, if there are 4 dosages for tablets indicated as D1, D2, D3, D4, then a dummy range of values can refer to each factor value such as 1 for D1, 2 for D2...4 for D4. the values 1...4 are known as dummy regressors. Sometimes a model can be generated using interaction terms as well.

The first step towards making a prediction is to estimate the coefficients. Let's suppose there are n observation pairs, each of which consists of X (predictor) and Y (response). The principal goal of applying this model is to find the estimates for $b_0$ and $b_1$ such that the linear model fits the available data well. An intercept $b_0$ and slope $b_1$ must be found so that the resulting line is as close to the actual data points. This closeness is measured using the least squares method. The least squares approach chooses $b_0$ and $b_1$ to minimise the RSS. RSS stands for residual sum of squares and can be defined as shown in Eq A.2.

$$RSS = {e_1}^2 + {e_2}^2 + .. + {e_n}^2 \tag{3.9}$$

Equivalent of RSS equation can be written as

$$RSS = (y_1 - \beta_0' - \beta_1' x_1)^2 + (y_2 - \beta_0' - \beta_1' x_2)^2 + .. + (y_n - \beta_0' - \beta_1' x_n)^2 \tag{3.10}$$

The least square approach chooses $\beta_0'$ and $\beta_1'$ to minimise the RSS using the following minimiser equations

$$\beta_1' = \frac{\sum_{i=1}^{n}(x_i - x')(y_i - y')}{\sum_{i=1}^{n}(xi - x')^2} \tag{3.11}$$

$$\beta_0' = y' - \beta_1' x' \tag{3.12}$$

### 3.5.2 Support Vector Regression (SVR)

The Support Vector (SV) algorithm is a non-linear generalisation of the Generalised Portrait algorithm developed by Vladimir Vapnik and his co-workers [82]. This algorithm is based on the supervised learning model and has its roots in statistical learning theory or VC theory that characterises the properties of learning machines to enable them generalising to unseen data. Support Vector Machines can be characterised by the usage of kernels, absence of local minima, sparseness of the solution and capacity control obtained by acting on the margin or on number of support vectors, large margin hyperplane, and usage of slack variables to overcome noise in the data etc. All these features were already known in the machine learning community since 1960's, however it was not until 1992 when all these features were put together to form the maximal marginal classifier, the basic Support Vector Machine. Support Vector Machines can be applied to both classification and regression problems.

Support vector regression is similar to any other regression technique. You give it a set of input vectors with associated response values and it tries to fit a model to predict the response given a new input vector [83]. Kernel methods apply some transformation on your dataset prior to the learning step and are used to capture non-linear trends in data. Using kernel methods you will have a hyperparameter that can be fine-tuned to get results. A simple decision function for linear SVM is defined as

$$f(x) = (w.x) + b$$

SVR works on the same principles as SVM to minimise the error and maximise the margin to separate the hyperplane; the difference is a margin of tolerance (epsilon) that is set in approximation to the SVM. In SVR the basic idea is to map the data x into a higher dimensional features space $F$ via a nonlinear mapping $\phi$ so the $f(x)$:

$$f(x) = (w.\phi) + b$$

Training the SVM means solving:

$$y_i - \langle w, x_i \rangle - b < \epsilon$$

$$\langle w, x_i \rangle + b - y_i < \epsilon$$

$x_i$ is the training sample with target value $y$. The inner product plus intercept $\langle w, x_i \rangle$ is the prediction for that sample and $\epsilon$ is used to set threshold so that all predictions have to be in range of $\epsilon$. SVM tries to adjust the hyperplane $\langle w, x \rangle$ by maximising the margin (1/2IIwII2) and minimising the training error. IIwII2 enforces flatness in the feature space. The constant $C$ determines the trade-off between the flatness of $f$ and the amount of deviation tolerated larger than epsilon. This corresponds to the $\epsilon$-insensitive loss function. The Polynomial kernel is one of the kernel methods used for the learning of non-linear models that represent the similarity of training samples (vectors) in a feature space over polynomials of original variables. So the same kernel trick is used as in SVR. The polynomial kernel with degree d is defined in Eq 3.13:

$$k(x, y) = (X^T y + c)^d \tag{3.13}$$

$X$ and $y$ are vectors in the input space, i.e. vectors of features computed from training or test data set. C is responsible to adjust the influence of higher and lower order polynomials. $K$ as a kernel corresponds to an inner product in a feature space based on some mapping $\phi$:

$$k(x, y) = \langle \phi(x), \phi(y) \rangle$$

The decision function in the non-linear case be defined as:

$$f(x) = w.\phi(x) + b$$

$f(x)$ learns the map from $\phi(x)$ to $y$, Where w can be completely described as a linear combination of the training patterns xi. Moreover the complete algorithm can be described in terms of dot products between the data. In a sense the complexity of a function's representation by SVs is independent of the dimensionality of the input space X, and depends only on the number of SVs. The difference to the linear case is that $w$ is not given explicitly. Also note that in the non-linear setting, the optimization problem corresponds to finding the flatness function in feature space instead of input space.

## 3.6 Revisiting the Related Research Goals

The main contributions of this chapter are reviewed by revisiting the research goals as well as requirements stated in Section 1.5 and 3.2 respectively.

Daleel offers an architectural design of a decision supports system integrated with a cloud brokerage framework. Daleel is designed to be a multi-criteria decision-making framework to assist the user in the selection of a suitable cloud provider and a virtual machine instance with regards to the application requirements and customer-specific constraints. The architecture allows module interaction with existing components related to interoperability and application management solutions.

The decision support system is designed to provide optimal decisions by making use of machine learning techniques. Machine learning methods are used to build learning models for predicting application's performance on different deployment setups. The model generation requires large amount of data to be used for the training and testing of learning methods. The data sets are collected containing information about the application deployment and resource monitoring details. The offered solution follows a modular design to support the specific tasks related to the intelligent decision making process. The designed solution is able to save all the information related to application, monitoring and decision solutions. Daleel follows an iterative and incremental approach to generate robust and generalised learning models that are equally effective across different applications.

## 3.7 Summary

This chapter has discussed the proposed approach of using machine learning for decision support system in a multi-cloud environment. The decision support system is designed as an integrated module within overarching cloud brokerage architecture to assist customers in making decisions at IaaS level according to defined requirements and constraints. The decision support module comprises of three major phases: *Analysis*, *Learning* and *Planning.* These phases run in an incremental and iterative manner to generate decisions starting from data collection to model generation. The *Analysis Phase* is responsible for gathering application profiling data that is comprised of details about application performance on different deployment setups. The generated application profiling data is then used by the *Learning Phase* to build a model for the prediction of application performance.

Model generation follows the traditional method of learning and deriving a model. A series of processes are executed in a circular fashion starting from model fitting, which requires the application of a series of statistical methods applied to analyse data and identify the basic structure of a learning model. This part of the process is the most important and time-consuming and any discrepancies can lead to inaccurate results. The fitted model is then trained on the training data. The trained model is further assessed for its accuracy using different statistical methods such as R2, cross-validation, and MSE. Model assessment is done on both the training and testing data. The Update fit phase is responsible to either increment the start of the process or to retrieve an output as a model estimate/prediction function f. A prediction function $f$ results from the iterative and incremental process of model generation.

In this study, the selected learning methods are MPR and SVR. There is always a trade-off between time to generate a learning model and to extract detailed knowledge about the true relationship of that model and so the polynomial regression and SVR are selected as representative methods to reflect the grey-box and black box nature of learning techniques. Regression methods are represented as inference methods providing a better understanding of the underlying association of response and predictor variables in a dataset. Once the detailed knowledge about inference relationship is available that knowledge can be used to generate other models using flexible approaches, such as SVR. Flexible approaches have the ability to take up any form to explain data variability, however, limited to explain the individual relationship of response and predictor. Chapter 4 elaborates, how these learning methods are tailored following the model generation steps for estimating prediction and inference functions.

Chapter 4

# The Traditional Learning Setting

## 4.1 Overview

Chapter 3 detailed the architecture of an intelligent decision support system integrated with cloud brokerage. The core intelligent part is based on three-phase modular architecture to support large-scale, back-end analysis to be fed iteratively into the model generation.

Chapter 4 focuses on the objective of creating learning models to capture application performance variations on different node configurations considering the black-box nature of IaaS resources. This chapter describes the experimental study and use of proposed approach for model generation using different machine learning methods that have already been described in 3.5. The experimental details include dataset information that are collected using real world applications and cloud IaaS. Details of model generation and evaluation of generated learning models are given in this chapter. This chapter also provides some experiential analysis and highlight potential advantage of using proposed approach.

## 4.2 Experimental Details

The objective of this experimentation is to validate if the decision support system is able to provide valuable behavioural insight about the application performance necessary for taking optimal decisions. A core output of the decision support module is the model generation to capture application behaviour and predict its performance according to deployment setup. A learning model can be generated using representative machine learning methods and following the model generation process of decision support module explained in Chapter 3.

Before getting into the description of application specific as well as generalised learning model generation, it is necessary to explain the experimental details related to applications and cloud instances that are used in this study. The datasets used for model generation and evaluation have been populated using representative applications running on a public cloud provider: Amazon EC2 instances. The reason for conducting extensive deployment experiments is to provide a sufficient amount of data for the model generation activities and specifically to avoid any synthetic data in this study.

### 4.2.1 Applications

Representative applications have been selected with different architectures and categories relating to their intensity of memory and CPU usage.

1. **VARD** is a tool designed to detect and tag spelling variations in historical text, particularly in Early Modern English [84]. The output is aimed at improving the accuracy of other corpus analysis solutions. VARD is a single threaded Java application that is highly memory intensive. It holds in memory a representation of the full text, as well as various dictionaries that are used for normalising spelling variations. VARD is considered a pre-processor tool to other corpus linguistic tools such as keyword analysis, semantic taggings and annotations etc.

2. **smallpt** is an open source, global illumination rendered application written in C++. Unbiased Monte Carlo path tracing is used for rendering the scenes. Smallpt is a multi-threaded OpenMP based application, categorised as CPU intensive. This application is a composition of different features such as anti aliasing, ray-sphere intersection and Russian roulette for path termination. This application requires a number of samples per pixel as input, which is considered as number of paths per pixel for rendering a scene. For this research we selected a box scene that is constructed out of nine very large overlapping spheres. The image is computed using numerical equations that solve the rendering equation. The Monte Carlo path tracing algorithm is used with Russian roulette for path termination. OpenMP is used to achieve parallelism for dynamic allocation of rows of the image to different threads where each thread is allocated to each available processor or core.

3. **Item Recommender** is an item-based recommemnder technique to recommend similar items, such as movies, to users based on the collaborative filtering technique that is using loglikelihood

similarity to identify similar items. The item recommender is a Java based program that compares users with critics of other users in the form of a rating from 0-10 and recommends movies by comparing the similarity amongst users. Item recommender uses the MovieLens dataset collected and made available by the GorupLensResearch, which targets the social computing research at the University of Minnesota [85]. The dataset is comprised of 10 million ratings of 10,000 movies by 72,000 users. The idea is to recommend movies to a user based on the preferences of other users. The program takes every other person who has reviewed at least 5 movies in common with the user and calculates the Pearson correlation between these 2 users. Based on the similarity between users, weights are calculated for the percentile ratings. The weights are then converted to numerical values to be assigned to the user's scale and finally the recommendations are listed after sorting.

**Summary**: These three applications have been selected to give a representative spread of application nature and types. The application characteristics are summarised in Table 4.1. By architecture, VARD is a single threaded, non-parallel application and requires lots of memory to hold data file for processing. The second application Smallpt is a CPU-intensive application following a multi-threaded and paralleled architecture. There is no external file requirements for this application. The third application Item Recommender is similar to VARD in terms of architecture. However it does not require the data file to be loaded into the memory. This application sits between two categories and has a medium CPU utilisation compared to smallpt.

### 4.2.2 Cloud IaaS

Extensive experiments are conducted on Amazon Elastic Cloud Compute (EC2) using a representative set of applications as mentioned earlier. The Amazon EC2 is the leading public cloud service provider with a 57% share of the IaaS market [86]. All instances used are 64-bit Ubuntu Linux of different capacities as shown in Table 4.2. Note that 'vCPU' indicates the number of virtual cores assigned to a VM. An 'ECU' refers to an EC2 Compute Unit ; Amazon does not advise how an ECU relates to physical processing speed; it only assures that it is a standard unit across its IaaS offerings. 'Price' refers to the hourly charge for running a VM of the referenced instance type.

Amazon provides a differentiated series of instance types, catering to different application needs (eg compute-intensive, memory intensive, I/O-intensive, and so on). Each series contains a number of instance types offering different setups of computational resources. We targeted the General Purpose

**Table 4.1:** Application specific properties (Application Vignette).

| **Properties** | **Application 1** | **Application 2** | **Application 3** |
|---|---|---|---|
| | **VARD** | **Smallpt** | **Item Recommender** |
| Category | Memory Intensive | CPU Intensive | Mix |
| External File | Y | N | Y |
| Multi-threading | N | Y | N |
| In-mem. Processing | Y | N | N |
| Parallel Processing | N | Y | N |
| File Size | 3kb | 0 | 123MB |
| CPU Utilisation | Low | High | Medium |
| Mem. Utilisation | High | Low | Low |

series T2 and M3 as well as the Compute Optimised series C4 in order to evaluate varying combinations of resource capacities over a relatively wide price range. Only on-demand instances are used for this experiment. These have no long-term commitments and are charged on a pay-as-you-go basis at an hourly rate. All instances are chosen to be located in eu-west-1 availability zone, hosted in Ireland. Amazon EC2 uses the Xen hypervisor to host the VM instances but does not provide the details of the scheduling algorithms used by the hypervisor. Some of the information, such as details of parallel workload on virtual machines, scheduling algorithms and how EC2 virtual cores are pinned to physical cores is not provided by the public cloud providers. So the users of infrastructure as a service cannot perceive any collocation or interference effect on their running application.

### 4.2.3 Experimental Setup

To collect a substantial amount of data, three representative applications are deployed on a subset of EC2 nodes, as explained in Section 3.4.1. Experiments are continuously repeated using the representative set of applications, over a period of seven days with a delay of ten minutes in between each pair of runs. The reason for the selected duration of the experiment is to investigate temporal variations. The Some of the parameters were fixed in order to reduce the dimensionality of experiments and evaluation. For example, two of the applications, VARD and Item Recommender, require external files as input. Therefore, the experiment was conducted with fixed input file sizes of 3kb and 123MB, respectively. While running the smallpt application the grid size to render the image was set to 200.

**Table 4.2:** The computational specification of EC2 instances.

| **Series** | **Node** | **vCPU** | **ECU** | **RAM** | **Storage** | **Price** |
|---|---|---|---|---|---|---|
| | | | | (GB) | (GB) | ($/h) |
| T2 (General Purpose) | t2.small | 1 | Var. | 2 | 20 | 0.026 |
| | t2.medium | 2 | Var. | 4 | 20 | 0.052 |
| M3 (General Purpose) | m3.medium | 1 | 3 | 3.75 | 4(S) | 0.070 |
| | m3.large | 2 | 6.5 | 7.5 | 32(S) | 0.140 |
| | m3.xlarge | 4 | 13 | 15 | 32(S) | 0.280 |
| C4 (Compute Optimised) | c4.large | 2 | 8 | 3.75 | 20 | 0.116 |
| | c4.xlarge | 4 | 16 | 7.5 | 20 | 0.232 |
| | c3.xlarge | 4 | 14 | 7.5 | 32(S) | 0.239 |

## 4.3 Model Generation

The model generation process is as described in algorithm 1 below and detailed in Section 3.4.2.

```
Input: (i) S(X, Y) where $X = x_i, ....x_n$
(ii) MLMethod = MPR or SVR
(iii) SplitRatio = R
Output: (i) SignificantPredictors = $\{X_x\}$
(ii) PredictionFunction = f
Function:1 $\rightarrow$ DataSplit (S,R)
Return $S_{Train}, S_{Test}$
Function: 2 $\rightarrow$ ModelGeneration ($S_{Train}, S_{Test}, MLMethod$)
Start ModelGeneration
Function:3 $\rightarrow$ ModelFitting($S_{Train}$, MLMethod)
Start ModelFitting
while satisfactory MSE is found do
    foreach $x_i \in x_1, ..., x_n$ do
        compute $y' = f(x) + \epsilon$
        compute coefficient values $\beta$
        compute F-statistics
        if F statistics and $\beta$ has a non zero value then
            $x_i.marked = significant$
        else
            $x_i.marked = nonsignificant$
        end
        Return $S_{significant}$
    end
    for all $x_i \in x_{significant}$ do
        compute $f'$ with polynomial or Kernel function
        remove outliers
        calculate VIF factor if required and applicable
        apply interaction terms if required and applicable
        apply parameter tuning if required and applicable
        compute $MSE_{Fit}$ for $f'$ where $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$
        compute $R^2_{Fit}$
        select $f'$ with highest $R^2$ and lowest MSE
    end
    select $f'$ with highest $R^2$ and lowest MSE
    Return $f'_{Fit}$
    End ModelFitting
end
```

```
Function:4 → ModelTraining ($S_{Train}$, $f'_{Fit}$ )
Start ModelTraining
for iteration $i$ = 1-2000 do
    Compute $f'_{Train}$ using re-sampling method $CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$ , where $k = 10$
    aggregateMSE = $MSE_{Train} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$
    compute $MSE_{Traint}$
    compute $R^2_{Train}$
end
Return $f'_{Train}$
End ModelTraining
Function:5 → ModelTesting ($S_{Test}$, $f'_{Train}$ )
Start ModelTesting
Compute $f'_{Test}$ using re-sampling method $CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$ , where $k = 10$
aggregateMSE = $MSE_{Train} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$
compute $MSE_{Test}$
compute $R^2_{Test}$
if $MSE_{Test} < MSE_{Train}$ then
    Return $f'_{Test}$
    save $f'_{Test}$ in Function Repository
else
    Call Function UpdateFit
end
End ModelTesting
Function:6 → UpdateFit()
Start UpdateFit
Call ModelFitting with new dataset
Call ModelFitting with different Learning method
End UpdateFit
End ModelGeneration
```

**Algorithm 1:** Algorithm for Model Generation

- **Input** $\rightarrow$ The Model Generation module of the *Learning Phase* gets input $S(X, Y)$ in the form of a profiled dataset generated by the *Analysis Phase*, where $Y$ is a response variable and $X$ is a set of multiple predictors. This thesis leverages the use of Multiple Polynomial Regression (MPR) and SVR to generate prediction functions.

- **Output** $\rightarrow$ The Model generation module aims to estimate the prediction function $f$ to capture the application's behaviour on different deployment setups. The response variable $Y$ for this prediction function $f$ is the application's execution time.

- **DataSplit** $\rightarrow$ After the machine learning method has been selected, the data must be split into **training and test sets**: **1)** a training set to be used for learning, and **2)** a test sample for assessment and model evaluation. For the current evaluation, we split the data set into a training and test set with a ratio of 60% and 40% respectively, which is above the conventional ratio of 70:30 percentage split [39].

- **ModelFitting** $\rightarrow$ Once the training and test sets have been finalized, the model must be fitted to the data as stated in Function:3 of Algorithm 1. This function has two steps. First, the significant predictors are marked. Second, the mathematical form of the function $f$ is identified applying various methods as explained in detail in Section 3.4.2-Application Vignette. The predictors in Table 4.3 are the significant predictors and are identified using a regression method, specifically multiple regression. Furthermore, the same predictors are used for generating an SVR-based model as well. The predictors are indicated as $X_i$, where $i$ represents the index number for the predictor variable. Section 4.3.1 and 4.3.3 elaborates the generated models which are presenting various forms of relationship that are derived using different subsets of significant predictors, as shown in Table 4.3.

- **ModelTraining** $\rightarrow$ involves training the fitted model function $f'_{Train}$ on the training dataset with fixed parameters that are calculated during the model fitting stage. Model training involves Monte Carlo type processing to train a model. This implies training a model in 2000 iterations and 10-fold cross validation is involved at each iteration.

- **ModelTesting** $\rightarrow$ The trained model is further assessed on the test dataset via the **Model assessment** step as explained in Function:5 of Algorithm 1. The model accuracy is assessed using p-value, $R^2$ and MSE values. The MSE result of test set lower than training set confirms

**Table 4.3:** Significant predictors.

| **Predictor ID** | **Predictor Name** |
|---|---|
| $X_1$ | ECU |
| $X_2$ | VCPU |
| $X_3$ | RAM |
| $X_4$ | Multithreading |
| $X_5$ | Load in memory |
| $X_6$ | Application Type |
| $X_7$ | External file requirement |
| $X_8$ | Parallel execution |
| $X_9$ | File size |
| $X_{10}$ | Day |

the satisfactory prediction results and that the model is not over-fitted. If the prediction model verifies satisfactory results, it is saved to the *Function Repository*.

- **UpdateFit** $\rightarrow$ The results of Model assessment are explored to determine if the model fit needs to be updated. If the results are not satisfactory, the next iteration can be started from model fitting. A different machine learning method can be used to generate the learning model for the same dataset and the whole process will re-start.

### 4.3.1 MPR Based Learning Models

Multiple models are derived for each application using the multiple polynomial regression method. These models are represented in simple mathematical form, where $X_i$ indicates the different significant predictors and ":" sign indicates the interaction between two terms. Table 4.3 provides details about the predictors $X_i$ where $i = 1..10$. The superscript number in the model indicates the degree of polynomial and all the prediction functions (learning models) are written using simple mathematical notation.

These models are generated following the traditional principle of applying machine learning where training and test data sets are drawn from the same data distribution. It means the learning models are generated and can be applied on a specific application for particular deployment setup.

**Learning Models for VARD**:

Learning models for the VARD application are indicated as Model 1, Model 2, Model 3 and Model 4. Multiple models indicate that prediction functions can be derived using different predictors and mathematical forms. However, some of the predictors are firmly present in each model. Deriving a range of models is a good approach to finding a generic model by indicating the use of common predictors and mathematical forms.

Model 1 (M1) can be described using the equation given in 4.1

$$Y = X_1^3 : X_4 + X_2^2 + X_3^3 : X_5 + X_6 \tag{4.1}$$

Similarly, Models 2 (M2), 3 (M3) and 4 (M4) are mathematically described as:

$$Y = X_1^3 : X_4 + X_2^2 + X_3^2 : X_5 + X_6 \tag{4.2}$$

$$Y = X_1^3 : X_4 + X_3^2 : X_5 + X_6 \tag{4.3}$$

$$Y = X_1^3 + X_3^3 + X_{10}^3 \tag{4.4}$$

The first two models are composed of using the same predictors and interaction terms but they vary in the degree of polynomial order. In contrast, Model 3 is composed of using different number of predictors as well as polynomial order. Model 4 is the simplest model using few predictors but a high order polynomial of degree 3.

Table 4.4 provides the model assessment of VARD prediction models. Model assessment is performed by using resampling methods and by comparing MSE values on the training and test data. The MSE values for the first three models are almost the same when compared to the last model that has the lowest MSE value amongst all models. The table clearly shows a reduction in MSE of test data when compared to the training data. This also indicates that the model is performing better on test data and is not an over fitted model. $R^2$ value for all these models depicts that models are capturing more than 96% of data variation.

**Table 4.4:** VARD's model assessment.

| **Model** | $R^2$ | **MSE (Training Set)** | **MSE (Test Set)** |
|---|---|---|---|
| Model 1 | 0.98 | 144.65 | 137.802 |
| Model 2 | 0.98 | 144.65 | 137.802 |
| Model 3 | 0.98 | 144.65 | 137.802 |
| Model 4 | 0.96 | 133 | 129 |

**Learning Models for smallpt**:

The following Models 1 (M1), 2 (M2) and 3 (M3) (described by the equations 4.5, 4.6 and 4.7) are generated for the smallpt application using the multiple polynomial regression method.

$$Y = X_1^3 : X_4 + X_2^2 + X_3 : X_5 + X_6 \tag{4.5}$$

$$Y = X_1^3 : X_4 + X_2^2 + X_3 + X_6 \tag{4.6}$$

$$Y = X_1^3 : X_4 + X_2^2 + X_3 : X_5 \tag{4.7}$$

Model 1 uses a large number of predictors and interaction terms compared to other models. Model 2 and Model 3 differ in the selection of predictors. Noticeably, all the models uses the same degree of polynomial. The model assessment of learning models for smallpt application is shown in Table 4.5. Reduced MSE values for test data set compared to training dataset confirms that the model is not over fitted. The highest value of $R^2$ for all three models confirms the accuracy of model in terms of capturing the data variability, which is 98% for all the models.

**Learning Models for Item Recommender**:

The following two models are generated for the Item Recommender application using polynomial regression. Model 1 uses an additional predictor for model composition as compared to model 2 and also uses a high order polynomial with interaction terms. Model 2 is considered a relatively simple model.

**Table 4.5:** Smallpt's model assessment.

| **Model** | $R^2$ | **MSE (Training Set)** | **MSE (Test Set)** |
|---|---|---|---|
| Model 1 | 0.98 | 9.8360 | 7.3716 |
| Model 2 | 0.98 | 9.8198 | 7.3716 |
| Model 3 | 0.98 | 9.8198 | 7.3716 |

Model 1 (M1) takes the following mathematical form:

$$Y = X_1^3 : X_4 + X_2^2 + X_3^2 : X_5 + X_6 \tag{4.8}$$

similarly, Model 2 (M2) can be mathematically described as:

$$Y = X_1^3 : X_4 + X_2 + X_3 + X_6 \tag{4.9}$$

The model assessment of learning models for the Item Recommender is shown in Table 4.6. MSE values for the test data are numerically less compared to the values of the training data and confirm that the model is not over fitted. In addition, these models are also capable of capturing 98% of data variability which is considered a considerable good value.

**Table 4.6:** Item-Recommender's model assessment.

| **Model** | $R^2$ | **MSE (Training Set)** | **MSE (Test Set)** |
|---|---|---|---|
| Model 1 | 0.98 | 23.901 | 22.718 |
| Model 2 | 0.98 | 25.226 | 22.067 |

### 4.3.2 Generic Learning Model

A number of models are generated for each of the application by applying multiple polynomial regression method. We have observed that all the models share some common predictors as well as polynomial order and based on the commonality a generic model can be derived. All of these models are considered equally good to capture non-linear patterns in the data set as explained by the $R^2$ value which confirms that models are capturing more than 95% of data variability. However, the MSE values for all these three applications are different and show slightly different levels of prediction accuracy.

Table 4.7: List of all learning models to identify similar terms to extract a generic learning model.

| **X** | **VARD** | | | | **smallpt** | | | **Item Recommender** | |
|---|---|---|---|---|---|---|---|---|---|
| | M1 | M2 | M3 | M4 | M1 | M2 | M3 | M1 | M2 |
| $X_1^2$ | N | N | N | Y | N | N | N | N | N |
| $\boldsymbol{X_1^3 : X_4}$ | Y | Y | Y | N | Y | Y | Y | Y | Y |
| $\boldsymbol{X_2^2}$ | Y | Y | N | N | Y | Y | Y | Y | N |
| $X_3^3$ | N | N | N | Y | N | N | N | N | N |
| $X_3$ | N | N | N | N | N | Y | N | N | N |
| $\boldsymbol{X_3^2 : X_5}$ | N | Y | Y | N | N | N | N | Y | Y |
| $\boldsymbol{X_3^3 : X_5}$ | Y | N | N | N | N | N | N | N | N |
| $\boldsymbol{X_6}$ | Y | Y | Y | N | Y | Y | N | Y | Y |
| $X_10^3$ | N | N | N | Y | N | N | N | N | N |
| $X_3 : X_5$ | N | N | N | N | Y | N | Y | N | N |
| $X_2$ | N | N | N | N | Y | N | Y | N | Y |

Generating a generic model requires the extraction of common predictors and similar non-linear patterns indicated by some mathematical form. In order to do this we build a table 4.7 that indicates such commonalities and similarities across the different learning models.

In Table 4.7, First column lists all the predictors which are part of different models. $M_i$ (from second column onwards) represents a model with respect to mentioned application. In the table, value **Y** confirms the presence of a predictor in $M_i$ model and **N** negates the presence of a particular predictor within a model. The common terms/predictors are highlighted in grey. This also indicates the similarity of non-linear patterns amongst different datasets of different applications. If any model uses higher degree terms then by default all the lower degree orders are also included in the model. This creates similarities in the pattern at different polynomial degrees. By extracting mostly used common terms/predictors in all the models can build up a generalised learning model. A generic model takes the following mathematical form:

$$Y = X_1^3 : X_4 + X_2^2 + X_3^3 : X_5 + X_6 \tag{4.10}$$

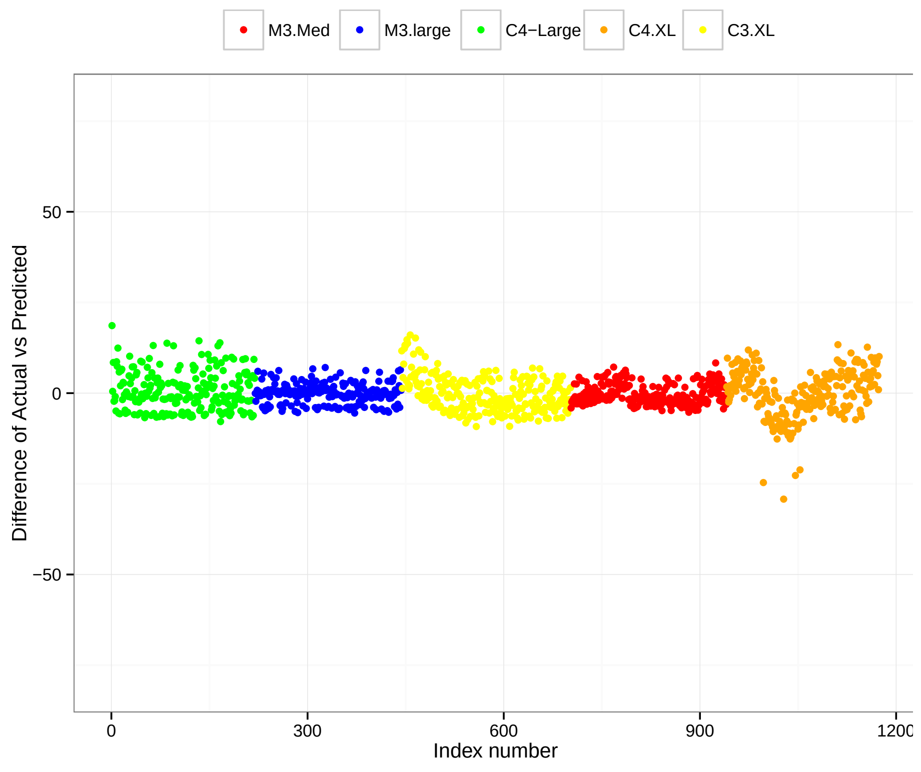


**Figure 4.1:** Plot of actual vs predicted values of itemRecommender

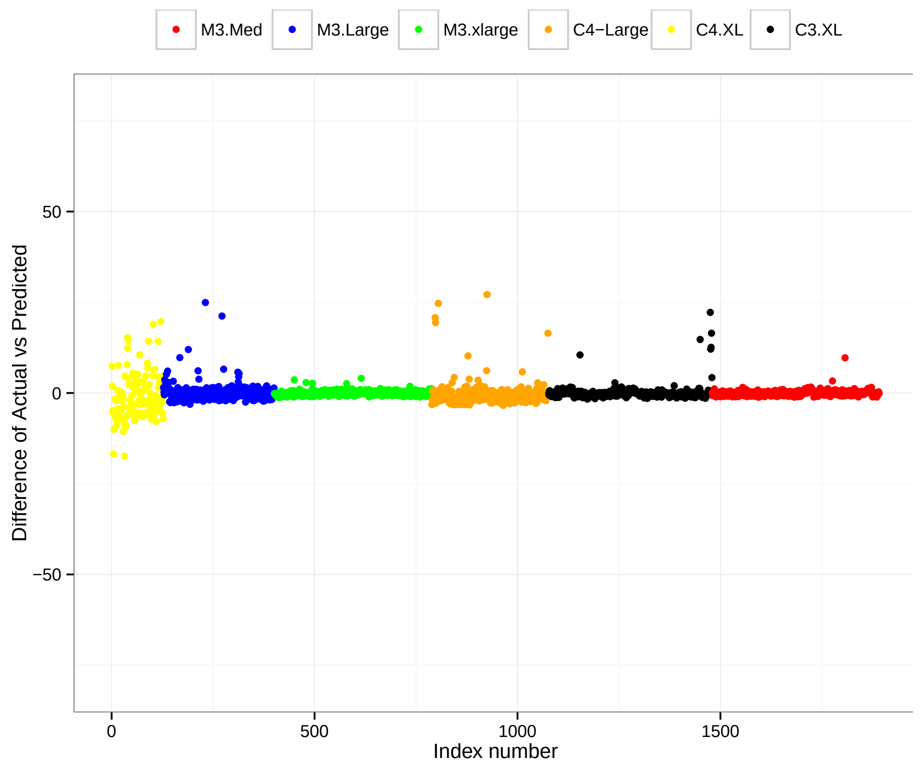


**Figure 4.2:** Plot of actual vs predicted values of smallpt

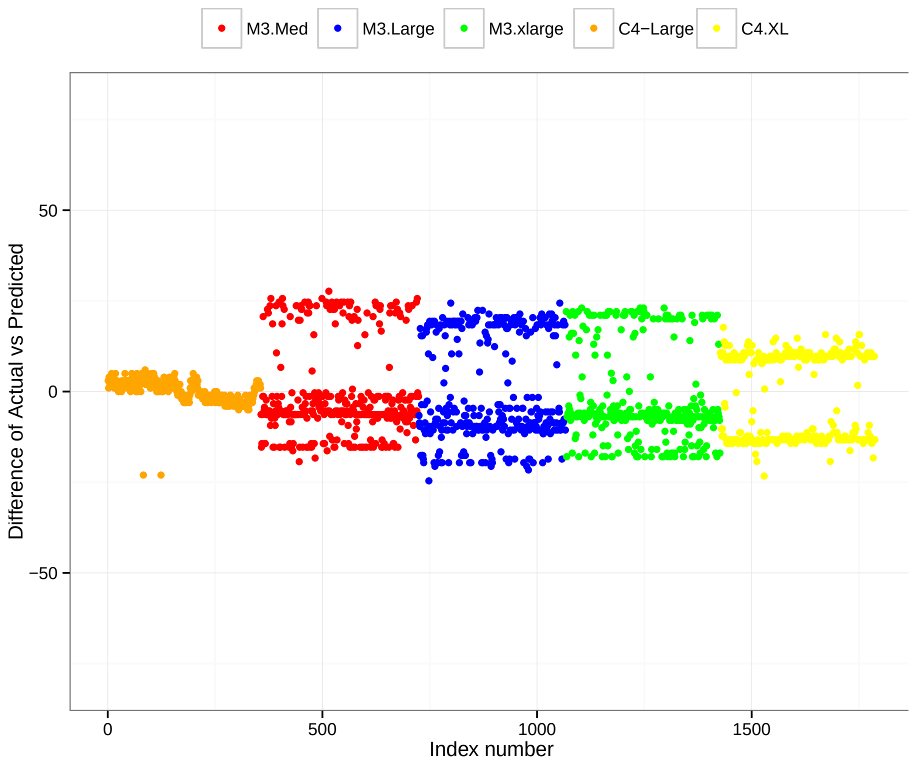


**Figure 4.3:** Plot of actual vs predicted values of VARD

**Table 4.8:** MSE values for generic model on test data.

| MSE (Smallpt) | MSE (Item Recommender) | MSE (VARD) |
|---|---|---|
| 7.5147 | 23.775 | 146.479 |

The following MSE values are computed when the generic model is applied on each of three applications, as shown in Table 4.8.The MSE values for the generic model are almost equivalent to the the application specific models and this confirms the accuracy of generic model.

The Figures 4.1, 4.2 and 4.3 display the difference of actual vs predicted values for three applications. The predicted values for response variable $Y$ are calculated by applying the generic model on test data of the three applications. In each of the graphs, the x-axis represent the index numbers of the values for the response variable in the test dataset. The y-axis represents the difference of actual and predicted values of the response variable, which is application execution time. In the graphs of smallpt and Item Recommender applications, the difference for most of the values are aligned around 0 and this indicates the good prediction accuracy. However, VARD indicates some spikes of difference which is mainly caused by too much variable and unpredictable performance of m3 series. The prediction accuracy for these two applications is also represented by MSE and $R^2$ values as shown in Table 4.8, 4.4, 4.6 and 4.5. Overall, the generic model as well as application specific models are performing equally well in terms of prediction.

### 4.3.3 SVR Based Learning Models

The second machine learning method that is considered for model generation in this thesis is SVR. SVR takes an input vector with associated response values and outputs a prediction function using kernel methods. Kernel functions apply data transformations to capture any non-linear trends in the data. SVR is a blackbox for end users as it does not provide some of the relationship details of predictors and response variable.

The same data sets which are used for model generation using polynomial method are used for SVR as well. However, instead of going through the same process of identifying significant predictors or blindly adding all the predictors, we preferred to use the same predictors which from the first place are result of first experimental phase with polynomial regression as a learning method. The rationale behind this is very clear: polynomial regression methods have provided sufficient insight about data relationships and importance of predictors. In this way, we can reduce the data dimensionality for running SVR based model generation, as only significant predictors are fed into the SVR model rather than the complete set of predictors. Reduced predictors can reduce the model complexity and a better result can be generated in less computational time. This would also save computation power and cost for running the complex computations.

At this point, instead of fitting an SVR model by exploring all possible kernel functions, existing knowledge can be used from any inference based methods, as we have seen that all the models that are generated shows a non-linear trend in data. The application of the regression method has already indicated the presence of a non-linear relationship that can be described by some polynomial function, thereby allowing the use of a polynomial kernel for SVR.

For the model generation most of the significant predictors are used in the SVR model that are identified using polynomial regression method, as stated in Table 4.3.

The SVR model is described in the following mathematical form along with all the tuning parameters and constant variables. The same model is applied on three of the representative applications and performing reasonably well in terms of prediction accuracy.

$$\begin{aligned}
&SVM(X_1(ECU) + X_2(VCPU) + X_3(RAM) + X_4(Multithreading) + \\
&\qquad X_5(Loadinmemory) + X_7(Externalfilerequirement) \\
&\qquad + X_8(Parallelexecution) + X_9(Filesize), Kernel = Polynomial, \\
&\qquad Coef = 0, Degree = 3, Learningrate = 0.3, cost = 1, \\
&\qquad Gemma = 1/d, Cachesize = 40, Fittedvalue = TRUE, \\
&\qquad Scale = FALSE, TerminationCriteria = 0.001, Epsilon = 0.1)
\end{aligned} \tag{4.11}$$

As a result of the various learning iterations, the following tuning parameters are derived for the SVR-based learning models. The SVR function requires a *Learning Rate* value that is chosen between 0 and 1. The *Learning Rate* in our model is 0.3. The *cost of constraints violation* is set to 1. The polynomial kernel is selected with *coef0* set to 0, and *Degree* to 3. The value of *Gamma* is set to 1/data dimensionality. The *Cachesize* is set to 40MB. The *Tolerance termination* criteria is set to 0.001. The *Epsilon* value indicates intensive loss function and is set to 0.1. The *Shrinking heuristics* is set to True. The models allow the inclusion of *fitted values* that can be used in each iteration to refine the results and this is set to true. The inclusion of fitted values is allowed in each iteration to further refine the model.

**Table 4.9:** MSE values for SVR Model on test data.

| MSE (Smallpt) | MSE (Item Recommender) | MSE (VARD) |
|---|---|---|
| 8.038 | 24.392 | 169.584 |

The computed MSE values for three applications using the SVR model are shown in Table 4.9. The MSE values for smallpt and Item Recommender are almost similar and for VARD a little higher, in contrast to the MSE values for application specific as well as generic models generated by the MPR method. The similarity at the MSE level for the generated models using both machine learning approaches confirms the model accuracy.

## 4.4 Experiential Analysis

We investigated the performance of running three representative applications on different EC2 instance types. The results of running VARD are summarised in Figure 4.4 where every dot represents the execution time of one run. Shorter execution times reflect a lower hourly rate over a full workload. There are several striking observations. First, contrary to intuition, m3.medium (a memory-rich instance) is of consistently poor performance. We also observe that c4.large surpasses both m3.medium and m3.large in performance. In fact it is on par with c4.xlarge, which is twice both in specification and cost. Overall, the T2 series offers by far the best value for money. A possible explanation is the

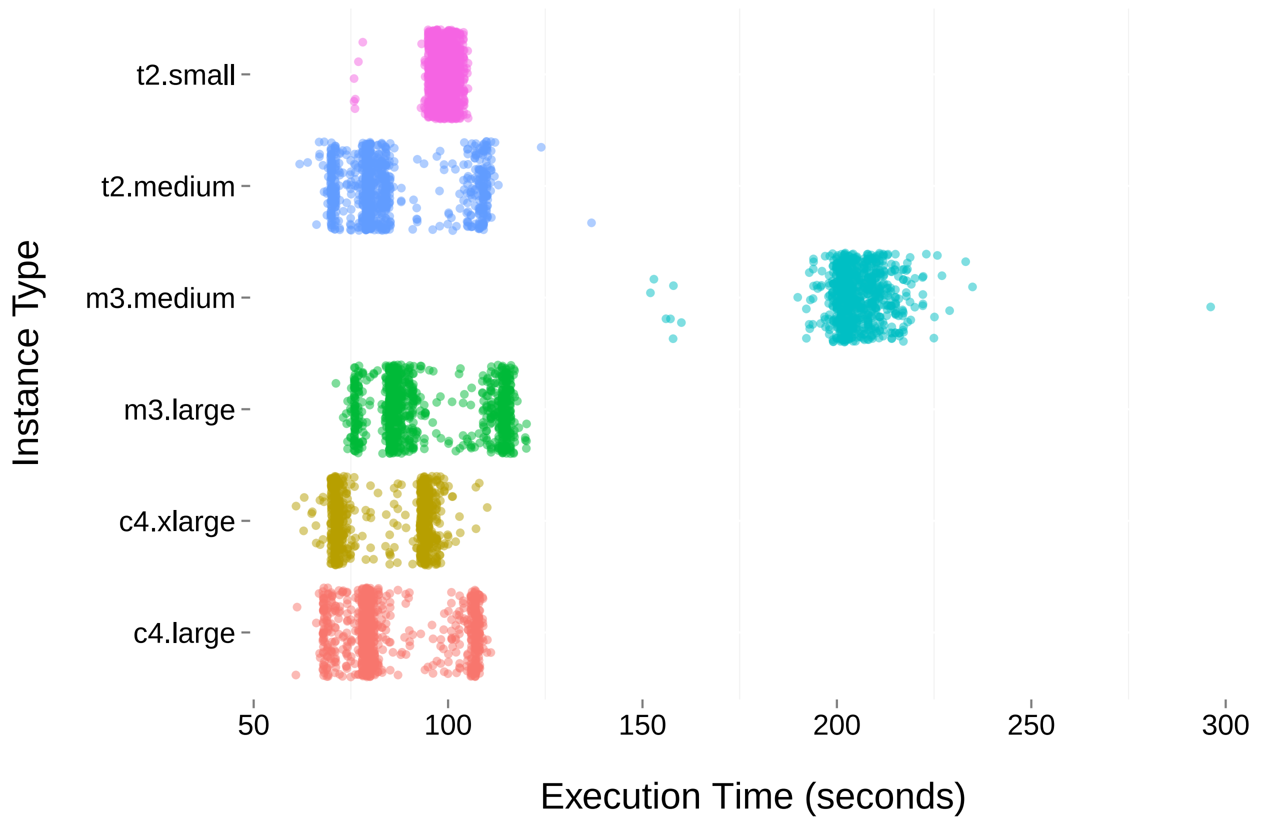


**Figure 4.4:** VARD execution time over different cloud instance types.

CPU Credits scheme, offered only on the T2 series, which enables customers to collect credits for idle instances and later spend them when full CPU utilisation is needed. T2 instances are thus good for applications that do not consistently fully use the CPU, but it also means that there is a degree of uncertainty associated with an application's performance that depends on its idle time.

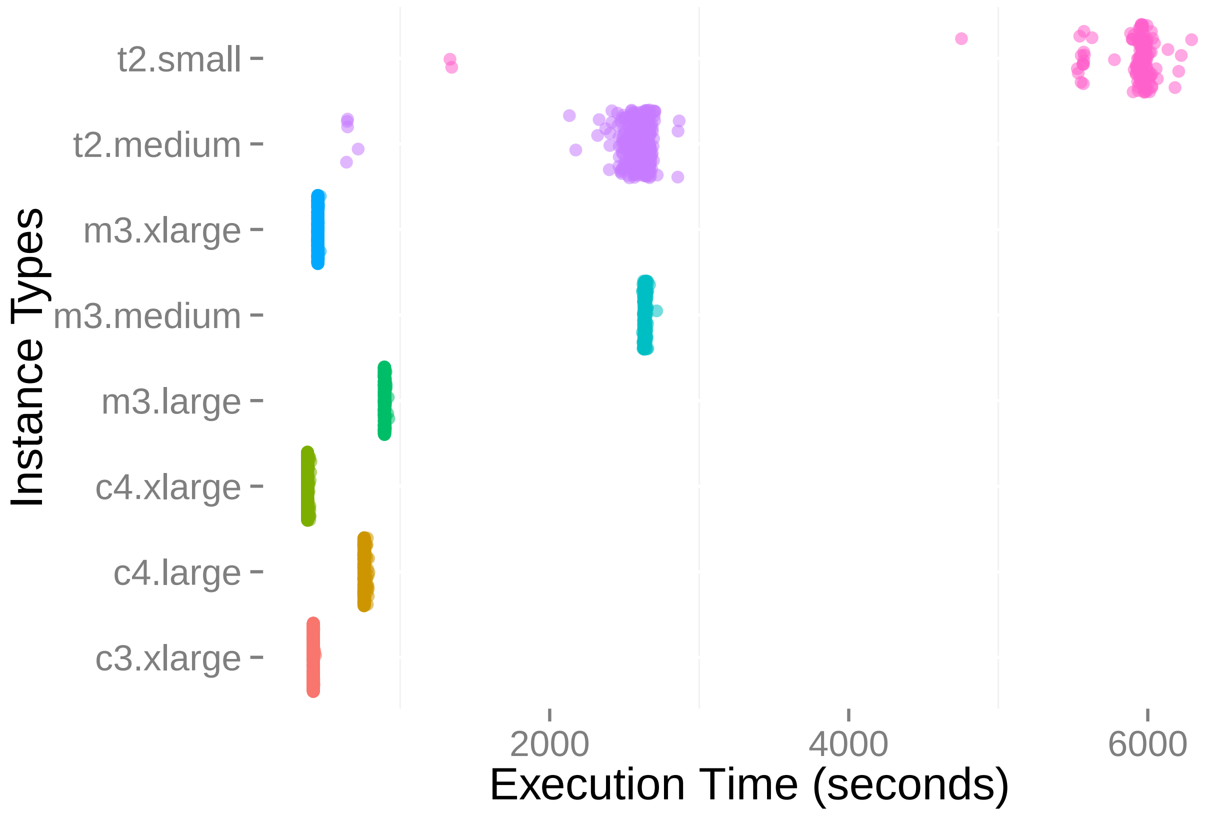


**Figure 4.5:** Smallpt execution time over different cloud instance types.

The results of running smallpt are summarised in Figure 4.5. Smallpt is a multi-threaded CPU-intensive application. In contrast to results of VARD, the performance of T2 series is low and the obvious reason is CPU capacity. Similar to the above results, m3.medium is performing poorly compared to other series. It is notable that m3.medium is performing similar to T2.medium which is far less in price. The performance of c3.xlarge and c4.xlarge are almost same. These instance types vary in price and are similar in vCPU capacity. An interesting observation is the same performance of both c3.xlarge and c4.xlarge, and this is also similar with c4.large, regardless of difference in offered ECU capacity. This indicates that the ECU capacity cannot be considered a standard measure of performance because the instance types are performing according to vCPU capacity. Another striking observation is the similar performance of m3.xlarge with c4.xlarge and c3.xlarge regardless of that m3 series is a recommended series for memory-intensive applications. This illustrates a similar performance of all .xlarge series.

The results of running Item Recommender are summarised in Figure 4.6. Item Recommender is similar to VARD in architecture but has a medium CPU utilisation compared to smallpt. To some

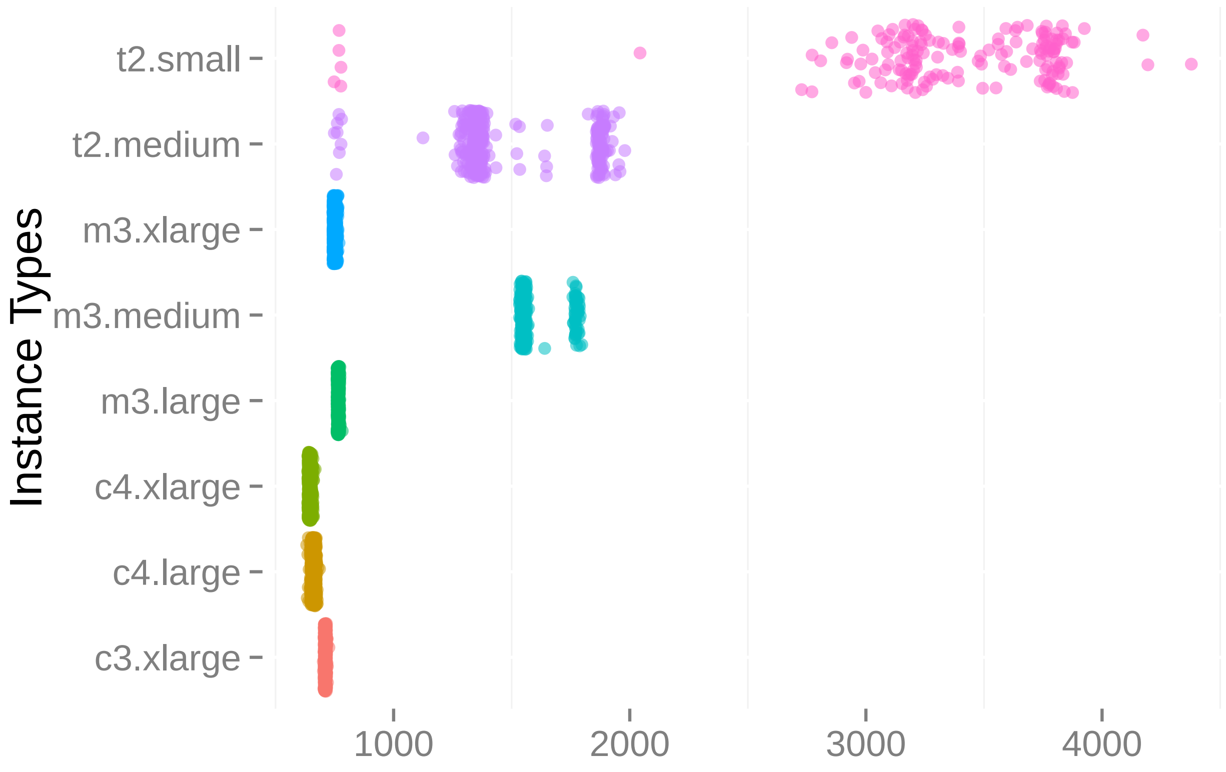


**Figure 4.6:** Item Recommender execution time over different cloud instance types.

extent, the results of Item Recommender are quite similar to smallpt. For example, the T2 series is not performing well and this reflects the CPU need of Item Recommender. Most importantly, m3.medium is again performing poorly regardless of its high price compared to T2 series. Surprisingly, c4.large sits between the .xlarge series in terms of performance, so a low price instance type is performing better than the instance type of high price. To some extent, instance types are performing according to vCPU power and the ECU unit is creating misleading information.

We now turn our attention to uncertainty in application performance due to the time at which they are executed. This is depicted by the box-plots in Figure 4.7. The T2 series offers the least RAM but exhibits the least variance in performance between the different days of the week. m3.medium VMs display application execution times that are fairly high albeit predictable: the median and quartiles show very little variation across the days of the week. m3.large also offers quite predictable performance across the week, with a narrow first quartile which is favourable. The two C4 instance types portray contrasting performances. c4.large is rather predictable with a steady median and right skewness (i.e. a very narrow first quartile). On the other hand, dispersion in the c4.xlarge instances is more towards the high end of application execution time with a median that is less regular: less left

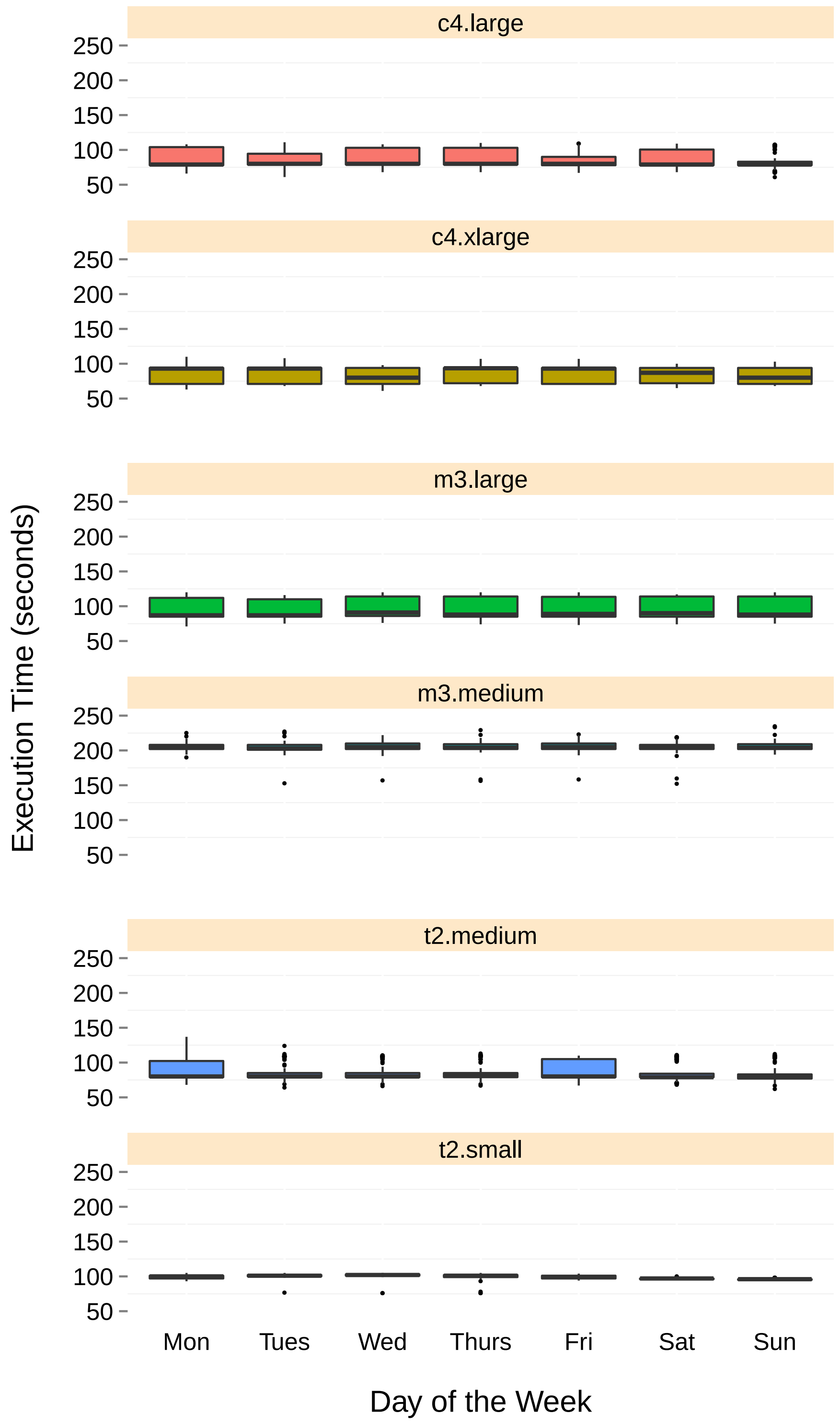


**Figure 4.7:** Dispersion of application execution time during all days of the week on different EC2 instance types. Notice that all graphs have the same y-axis range apart from m3.medium.

skewness is observed on Wednesday, Saturday and Sunday. This could be down to different reasons such as demand from other users, the provider's resource sharing algorithms, and the provider's energy efficiency policy. These are difficult attributes for us to ascertain from the outside. Nevertheless, we detect certain regularities that helps us determine the predictability of application performance at different times.

## 4.5 Summary

Decision making in cloud environments is not a trivial task especially when it comes to the selection of IaaS, keeping in view the QoS constraints. Traditional machine learning methods can help ease the difficult decision making process of choosing the right resource for the application and the handling of constraints as well. The basic principle behind the learning process is quite straight forward: build a model, train the model using collected data and use the model for predicting unseen data. These steps are followed to build different learning models for the three representative applications which are VARD, smallpt and Item Recommender. Data collection is done using extensive experimentation of deploying these applications on different instances of Amazon EC2. Multiple learning models are generated using two machine learning algorithms such as polynomial regression and SVR. A generic model is derived out of all regression-based learning models build for individual applications. Table 4.10 shows the MSE values for two models applied on the test data sets of three applications. The MSE values for both SVR and polynomial regression models are close enough to indicate that both models are performing equally well.

We also investigated how variable the performance obtained from different IaaS settings could be, making the execution of a simple application rather uncertain, as explained in Section 4.4. This demonstrates that public IaaS offerings are to a great extent black boxes. First, selecting instance types solely based on their advertised resource specifications is not necessarily optimal. Second, selecting which day of the week to run an application could result in significant variation in performance. Third, choosing a wrong deployment setup can lead to high computational cost.

**Table 4.10:** MSE values for SVR and Polynomial model.

| Learning Method | MSE (Smallpt) | MSE (Item Recommender) | MSE (VARD) |
|---|---|---|---|
| SVR | 8.038 | 24.392 | 169.584 |
| Polynomial regression | 7.5147 | 23.775 | 146.479 |

### 4.5.1 Potential Benefits of Daleel

Daleel equips a cloud customer with evidence-based knowledge of an IaaS setup specification that is optimal for the customer's particular application. The integrated architecture of the intelligent decision support system with a cloud brokerage framework supports the provisioning of realistic and application driven decisions that require necessary insight about the application behaviour on different deployment setups. Daleel supports the development of learning models to predict application performance and leverages the use of machine learning algorithms to generate learning models in order to capture the application's behaviour on different deployment setups. Such integrated solutions can be offered as a brokerage service to assist customer for making optimal deployment decisions by abstracting away the underlying complex methods and cumbersome comparison tasks.

### 4.5.2 Discussion

The current research work follows the traditional approach for applying machine learning methods. Total learning time includes time for data collection as well as model generation. A large amount of cost and time is involved in the first phase of our experiments where we tried to collect sufficient data to explore any temporal variability even at the day level. Further to that, we were interested to generate multiple learning models using different machine learning algorithms while keeping the same observational criteria for the purpose of evaluation. Each of the experiment had a data collection cost of \$153.888 involving 8 virtual machines belonging to different series and price range on Amazon EC2 running 24 hours for 7 days. On the other hand, learning cost varies for SVR and MPR. MPR is a representative of inference based methods and is used to get a better understanding of the actual relationship between a response variable and predictors. This learning method requires human intervention at various stages, as explained in Section 3.4.2, therefore requires more learning time. This learning effort, however, is reduced to some extent with the help of R-markdown script which can generate and display most of the important finding as well as visual graphs in nearly 30-60 minutes (depending on different data size, number of methods, iterations and computational speed). Based on the code generated findings, users with a different range of expertise can take few hours to a couple of days to generate a reasonable prediction model. These models are advantageous in certain aspects: Get a robust knowledge about the underlying relationship of response and predictors to generate a concrete set of outcomes. Moreover, use that knowledge to feed into complex machine learning methods to enhance the level of understanding. SVR is a representative of complex models which do not start with initial assumptions and the learning function is free to adopt any mathematical form. This, however, requires a lot of training time to adjust its parameters and does not clearly describe a transparent relationship of response and predictors at the end. This research follows the same approach of extracting useful information (from MPR based models) and to use it within complex learning methods (SVR based models) as explained in Section 3.5. By following this approach the training time of generated models ranges from a second to a maximum of 1 minute considering 10-fold cross-validation. This evidently describes a trade-off between the level of understanding and learning time.

A common assumption in a traditional learning setting is that the test and training data set are drawn from the same distribution and if the distribution changes then the lengthy process of rebuilding the model starts from the first step. Furthermore, the model derived for one type of distributional base data might not produce effective results for a different distribution. The change in distribution could be due to different applications or different cloud providers or virtual machines. This may result in having to repeat the approach from scratch by data collection. This leads us to think about generating a learning model be trained to produce an equally effective result with different distributional data. We tried resolving this matter by creating a generic model (Section 4.3.2) which can work equally effective on representative applications, yet not tested on different cloud providers. Conducting such experiments is still time-consuming and requires a cost for data collection. At this point, further challenges come into view from the perspective of cost-effectiveness that give rise to questions such as:

1. How can the cost and time be reduced when applying the machine learning technique?
2. How can we make our solution viable across different applications and cloud providers?

Our first intuition to answer above questions leading us to think about re-usability of existing knowledge that has been generated while creating learning models for different applications.

# Chapter 5

# The Transfer Learning Setting

## 5.1 Introduction

Chapter 4 has demonstrated that machine learning can play a vital role in designing an intelligent decision support system. Moreover, it provided the traditional principle of generating application specific as well as generic models using two machine learning methods, i.e. polynomial regression and SVR. The generated models are able to capture application behaviour on different deployment setups in order to make application-driven decisions. The chapter also examined the efficiency of the learning techniques, recognising that machine learning can impose significant training overhead.

Chapter 5 investigates enhanced learning techniques in order to make our proposed decision support system more efficient in terms of cost and time thus addressing the third research goal as stated in Chapter 1 and recalled here.

*"The development and evaluation of an efficient decision-making method integrated with the established decision support system to reduce the learning and decision-making cost and to making it more cost-effective for use in cloud brokers."*

In particular, this chapter introduces a novel two-mode transfer learning scheme leading to substantial reduction in the training overhead. The chapter also details the fundamentals of transfer learning technique and methods of transferring knowledge across domains. Furthermore, it explains how the two-mode transfer learning scheme is used to enhance the capability of our decision support system to make it more cost-effective for multi-cloud brokers. This transfer learning aided decision support system is evaluated using different applications and two public cloud providers, namely AWS and GCE.

## 5.2 Transfer Learning

Traditional machine learning is characterised by training data and test data having the same input feature space and the same data distribution. When there is a difference in data distribution between the training data and test data, the results of a predictive learner can be degraded [87]. In certain conditions, obtaining training data that matches the feature space and predicted data distribution characterisation of the test data can be difficult and expensive.

Transfer learning is used to improve a learner from one domain by transferring information from a related domain. An example from real-world but non-technical experience is quoted here to express the feasibility of transfer learning. Consider an example of two people who want to learn to play the piano. One person has no experience of playing any music and the other person has an extensive background of playing the guitar. The person with knowledge of the guitar will be able to learn piano more efficiently by transferring the previously learned musical knowledge to the task of learning to play the piano. The piano and guitar are considered sub-domains of a music domain, so common knowledge can be utilised to train the target learner based on the domain similarity [88]. As such, transfer learning provides a mechanism to solve new problems faster or improve their solution by using previously learned knowledge.

By definition, transfer learning aims to extract knowledge from one or more source domain and source task and apply that knowledge to achieve a target task in the target domain [88, 89]. The process of achieving the target task by learning from the source domain and source task is depicted in figure 5.1.

In a traditional statistical setting, we define a domain as a collection of two components: a feature space $X$ and a marginal probability distribution P(X), where

$$X = \{x_1, x_2 ..., x_n\} \in \mathcal{X}$$

Given a source domain $D_S$ and source task $T_S$, a target domain $D_T$ and target task $T_T$, transfer learning aims to help improve the prediction function of $D_T$ using the knowledge in $D_S$ and $T_S$ where $D_S = D_T$ or $D_S \neq D_T$ or $T_S = T_T$ or $T_S \neq T_T$ [88].

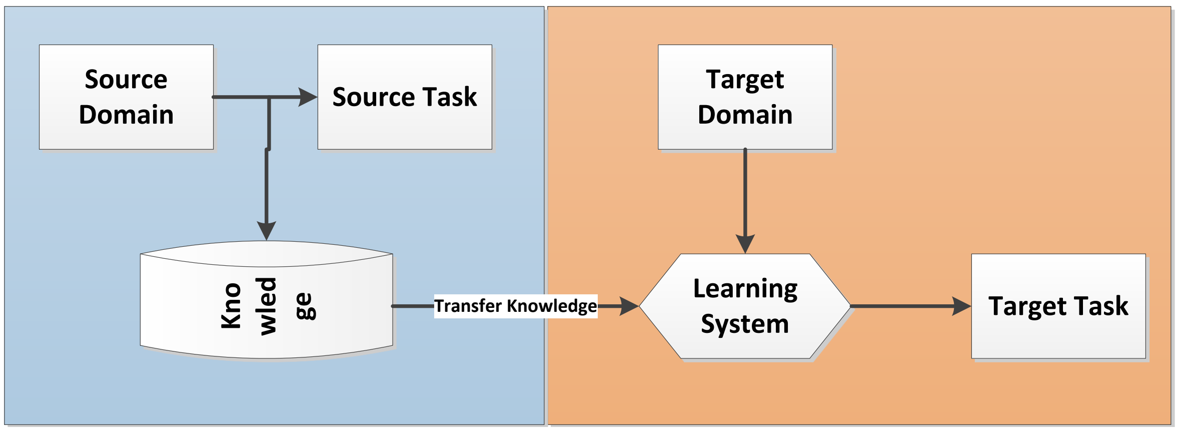


**Figure 5.1:** Fundamental approach of Transfer Learning.

Transfer learning has been applied to many real-world applications including image classification [90, 91, 92], sentiment classification [93], human activity classification [94], software defect classification [95] and multi-language text classification [96, 97, 98]. In order to transfer knowledge, both the source and target domains should have some similarity. Rosentein et al. emphasised the relatedness of the source and target domains in order to avoid any negative knowledge transfer [99]. Weighted function of conditional probability, flat naive bayes and KL-divergence for rank of domain are some of the methods used to measure the relatedness of the source and target domains for classification based transfer learning [100, 99, 101].

### 5.2.1 Transfer Learning Techniques

Transfer learning can be categorised into two subtypes, inductive transfer learning and transductive transfer learning [88].

#### Inductive Transfer Learning

Inductive transfer learning aims to improve the learning of target predictive function with the help of source domain $D_S$ and source task $T_S$, considering that the source and target domains are the same ($T_S = T_T$) but the tasks differ ($T_S \neq T_T$) [88]. This type of transfer learning can further be extended in two modes: *multi-task learning* and a *self taught learning*. Multi-task learning refers to the case where a source domain has multiple source tasks and each of these tasks is achieved simultaneously using an existing labelled source dataset [102, 103, 104]. In self taught learning, a task is achieved using unlabelled source data due to the absence of valid labelled data [105]. This type of learning runs in various iterations and tries to label the data to make it usable for further learning.

#### Transductive Transfer Learning

Transductive transfer learning aims to improve the learning of target predictive function with the help of $D_S$ and $T_S$, where $D_S \neq D_T$ but the tasks are the same ($T_S = T_T$) [88]. This is also referred to as domain adaptation [106]. The difference between the source and target domain appears either due to a varying feature space or a different marginal probability distribution. Transductive transfer learning can be categorised as unsupervised learning, although this condition can be relaxed with the inclusion of a little amount of target data to give the model an idea about the distribution of the target domain. Such auxiliary data helps in boosting the performance of the prediction model. Such transfer learning methodology is referred to as semi-supervised transfer learning or semi-supervised transductive transfer learning, where the auxiliary target data is available at the training time and this data is not part of the test data [107, 108].

### 5.2.2 Approaches to Transfer Learning

In a transfer learning setting, the most critical question is to identify the type of knowledge that can be transferred from a source to a target domain. Different ways of transferring the knowledge between two domains have been explained in [88, 89]. We briefly review them.

1. **Transferring knowledge of instances**

   Instance knowledge transfer is applied by re-weighing some portion of source data to be used in the target domain and iteratively measure the model fitness for target task learning [109, 110, 111, 112, 113]. MSE is considered one of the fitness measures for model assessment. Instance knowledge transfer seems quite intuitive considering the fact that the source and the target domains are quite similar in data distribution, and the use of the source data (instance) at training time can have a positive impact on learning a target task for the target domain. Even with a slight difference in distributions of target and source domain, one can still make use of the source data for learning a target task. Despite the fact that this method of knowledge transfer seems straight forward, misleading data examples from source data can lead to negative transfer of information [99].

   A suitable example for this type of knowledge transfer is when there is a lack of high-quality training data or collection of the training data is very expensive. In this case, the training and test data from some previous similar task can be used as a training set to achieve the target task.

2. **Transferring knowledge of feature representation**

   This type of knowledge transfer requires identification of good features that can reduce the differences between source and target domains in order to minimise the model error and domain divergence [105, 108, 114, 115, 116]. More effective results can be generated by increasing the weights on features that can fairly represent the target domain and are part of the source domain as well. However, this depends entirely on being able to select the right features and, if this is not achieved, the end result can be a poorly fitted model.

   Suppose that the source and the target applications vary in category; for example, one is memory-intensive and the other CPU-intensive. A learning model for the target application can be generated by extracting or using those features that reduce the differences between the source and target applications.

3. **Transferring knowledge of parameters**

   This approach transfers the parameter knowledge with an assumption that the source and target task share some parameters or prior distributions of the hyper-parameters of the models [104, 117, 103, 118]. Hence, the details of the learning model (source task) are transferred from the source to the target domain, such as, the degree of smoothness, kernel information, learning rate, learning constants, weights in the loss function, and so on. This type of knowledge transfer can work well in the case when both source and target domains are different in terms of data distribution as well as feature space, but the objective task is the same for both.

4. **Relational Knowledge transfer**

   This knowledge transfer approach deals with the transfer learning problems in relational domain [119, 120, 121]. This approach does not assume that the data drawn from each domain be independent and identically distributed, and can be represented by multiple relations, such as networked data and social network data. The statistical relational techniques are widely been used to solve this problem. This knowledge transfer technique builds mapping of the relational knowledge between the source and target domains.

## 5.3 Transfer Learning-aided Decision Support System

### 5.3.1 Motivation

In principle, transfer learning can be applied when the training and test data sets are drawn from different distributional data. This characteristic contrasts with traditional machine learning. In the traditional machine learning setting, if the data distribution changes, a learning model needs to be rebuilt from scratch starting from data collection. Therefore, in a real-world scenario repeating this activity is not efficient and incurs both additional time and cost. This is the situation our intelligent decision support system is faced with. The different applications and varying deployments setups lead to a change in data distribution as well as feature space. Transfer learning has a potential to deal with the challenge of a model generation when data distribution or feature space differ between source and target domain.

### 5.3.2 Problem Formulation

The decision support system explained in Chapter 3 required a learning model to predict application performance on different virtual machines in order to make an optimal deployment decision. The learning models are generated for every application following the traditional principle of machine learning, which is a time-consuming job. Moreover, the learning process also requires a huge amount of data collection for training and testing purposes, as described in Chapter 4. Repeating the same process for every new application incurs significant learning time and model generation cost. This causes a decrease in efficiency due to learning overhead, which decreases the overall efficiency of the decision support system.

In this context, efficiency can be improved by reducing the training overhead and the learning cost of model generation thus addressing the third research question stated in Chapter 1 and described as:

*"The development and evaluation of an efficient decision-making method integrated with the established decision support system to reduce the learning and decision-making cost and to making it more cost-effective for use in cloud brokers."*

### 5.3.3 Overview of Proposed Solution

The proposed approach is designed in the context of the cloud computing domain, where an intelligent decision support system is assisting customers in making optimal deployment decisions. The core objective of this approach is to provide efficient decision-making by reducing the training overhead to generate a learning model. There is a strong match between the properties of transfer learning and one of our main objectives about enhancing the efficiency of intelligent decision support systems. The proposed solution, therefore, targets the use of the transfer learning technique to help achieve such a goal.

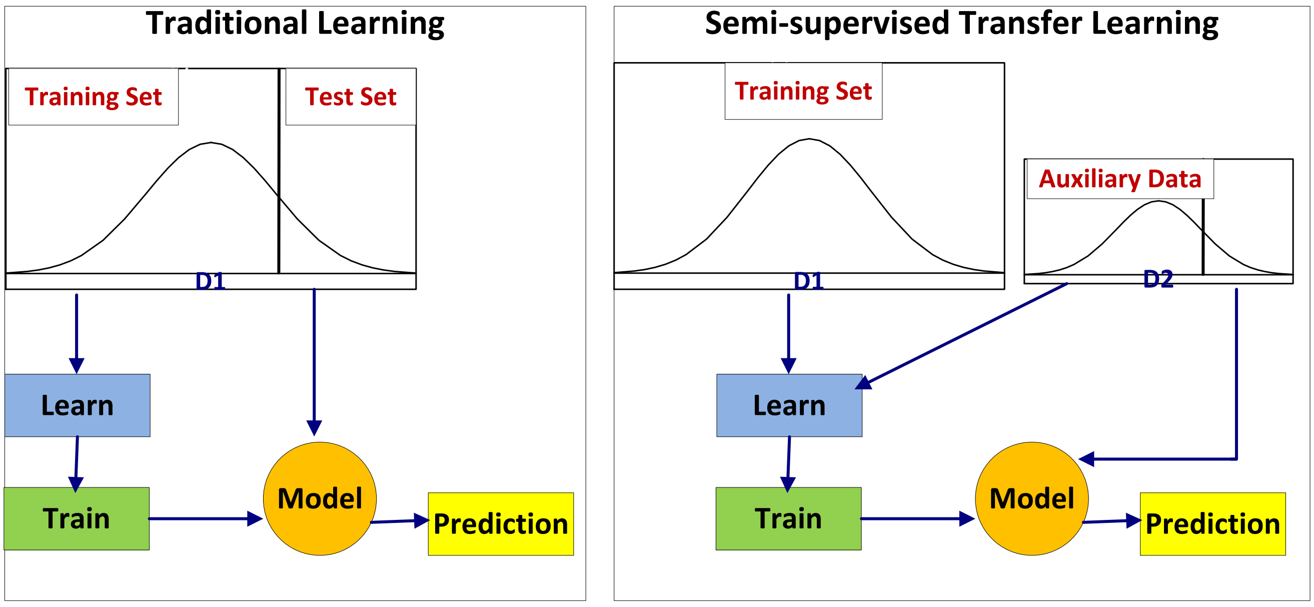


**Figure 5.2:** Use of auxiliary data in the transfer learning technique.

Transfer learning is used to generate a learning model for the target domain using the existing knowledge of the source domain, considering that the data distribution or feature space differs in both domains. In relevance to the target objective this thesis answers the following key questions:

1. How to identify the source and its learned data which can give the best performance to realising the target task?
2. What type of source knowledge can contribute towards achieving our goal and how to transfer that knowledge?

3. How to avoid negative knowledge transfer?

4. Which machine learning methods can be used under the transfer learning methodology.

In this research, the applied method of transfer learning is the semi-supervised transductive transfer learning method that allows the contribution of auxiliary target data for model generation. The motivation for using the semi-supervised approach is its ability to learn with a little amount of labeled data. This allows the reduction of the required training data for the target domain, which is one of the key concerns of learning efficiency. The auxiliary data contributes at the model generation phase from the target application. The complete training data set may be composed of the source instance data as well as the target instance data. Figure 5.2 provides details of a model generation and prediction process for both traditional learning and transfer learning. The left-hand side of the figure explains the traditional principle of a model generation where test and training data are drawn from the same distribution. In contrast, transfer learning makes use of learned data along with some auxiliary data for model generation. The right-hand side of the figure explains this process where learned data is coming from a domain D1, while domain D2 is contributing by providing auxiliary data. This is the approach our proposed solution is based on.

## 5.4 Daleel in the Transfer Learning Setting

This chapter presents a novel contribution of this thesis, a **two-mode transfer learning scheme**, which is designed to satisfy the goal of enhancing the learning efficiency. This scheme is designed using the fundamental principles of transfer learning and integrated with the existing architecture of Daleel. Hence, supported by the Learning Phase as shown in Figure 5.3. This scheme requires auxiliary data from the target domain and works in accordance with the base learner and similarity of source and target domain. The two-mode transfer learning scheme is derived from an extensive experimental analysis involving two public cloud providers and three representative applications. No synthetic or simulated data is being used at any stage of developing or evaluating this scheme.

This section details about the functionality of two-mode transfer learning scheme along with its important modules related to auxiliary data, the base learners, and the similarity measurement. It also describes how the existing intelligent decision-making module compliments the use of the proposed approach.

In this regard, only the relevant modules of Daleel are discussed which support the functionality of two-mode transfer learning scheme: these are *Analysis Phase* and *Learning Phase*. The internal architecture of Analysis and Learning phase is presented in Figure 5.3, where Learning Phase supports the transfer learning process.

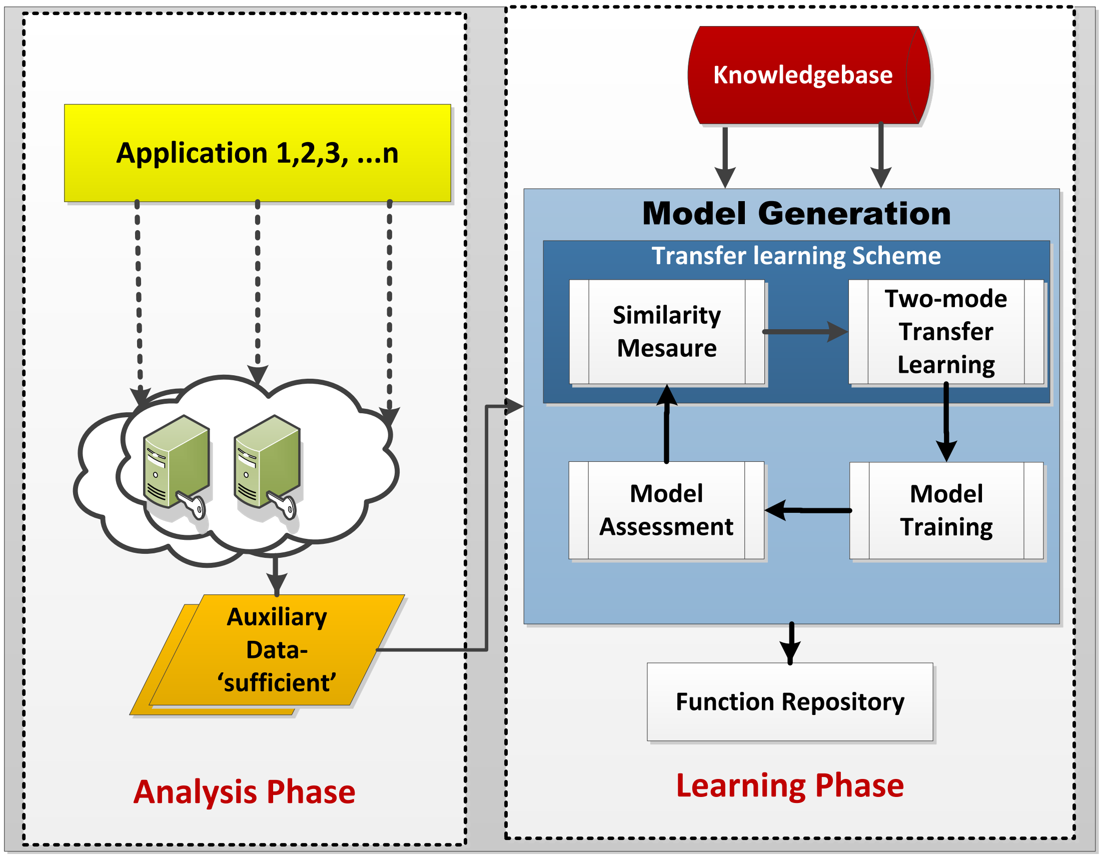


**Figure 5.3:** Daleel's decision support architecture in knowledge transfer setting.

As mentioned earlier, this research is following the semi-supervised transfer learning approach so the proposed methodology requires auxiliary data from the target domain. The *Analysis Phase* is responsible to collect auxiliary data. The function of the *Learning phase* is provisioning of the learning models. The model, however, is not generated from scratch by following the long steps of model fitting, as described in Chapter 3. Rather, the model is generated using the proposed *two-mode transfer learning* scheme as shown in Figure 5.3.The generated model is then trained using the transferred knowledge as well as the sufficient amount of target application data. Model assessment is performed on the generated model and if the output is not satisfactory, the process re-starts by fetching a new application from the knowledgebase. The test data set for the model assessment is comprised of the target application's data only. Finally, the accepted model is saved in the Function Repository to be used by the *Planning Phase*.

The complete model generation process is briefly explained via flow chart as stated in Figure 5.4.

1. The *Analysis Phase* is responsible to collect a 'sufficient' amount of data for the target application.

2. The *Learning Phase* receives data and starts the similarity measurement process.

3. The source application(s) is tagged according to the similarity output.

4. The two-mode transfer learning scheme starts transferring the knowledge according to the selected scheme based on the similarity result as explained in Algorithm 4. If, similar or partly similar applications are identified and the transferred learning method is SVR then the learning model is generated using the transferred knowledge by following the *Transfer-All* scheme. In contrast, if the applications are not similar and the learning method is still SVR, the *Transfer-Model* scheme is responsible to generate a learning model for the target application. On the other hand, if the learning method is MPR, a learning model is generated by following the *Transfer-Model* scheme.The detail of transfer learning approaches are already explained above in detail.

5. The generated model is then trained using the transferred knowledge as well as the sufficient amount of target application data. Most importantly, in a *Transfer-All* scheme the training data set is composed of two different distributional data coming from the source and the target domain.

6. Model assessment is performed on the generated model and if the output is not satisfactory the process re-starts from step 3. The test data set for the model assessment is comprised of the target application's data only.

7. The accepted model is saved in the Function Repository to be used by the *Planning Phase*.

Now, we explain the detailed functionality of the two-mode transfer learning scheme and its supporting modules such as the auxiliary data and the similarity measure.

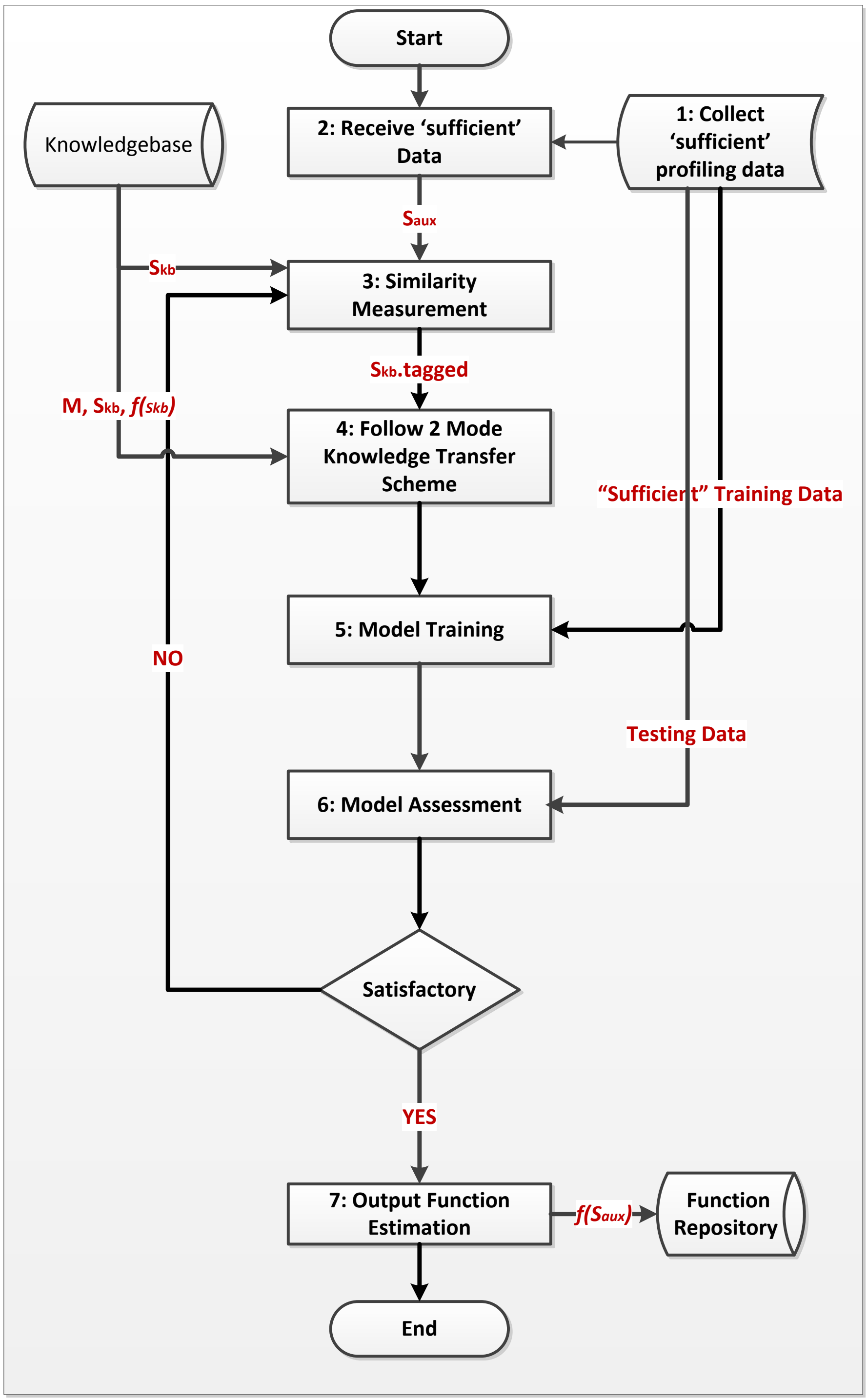

Start
Knowledgebase
2: Receive 'sufficient' Data
1: Collect 'sufficient' profiling data
Saux
Skb
3: Similarity Measurement
Skb.tagged
M, Skb, f(Skb)
4: Follow 2 Mode Knowledge Transfer Scheme
"Sufficient" Training Data
5: Model Training
NO
Testing Data
6: Model Assessment
Satisfactory
YES
7: Output Function Estimation
f(Saux)
Function Repository
End

**Figure 5.4:** Flow chart of the *Learning Phase* in transfer learning setting

## 5.4.1 Auxiliary Data

As mentioned earlier, application of the semi-supervised transductive transfer learning requires presence of the auxiliary data for the model generation.

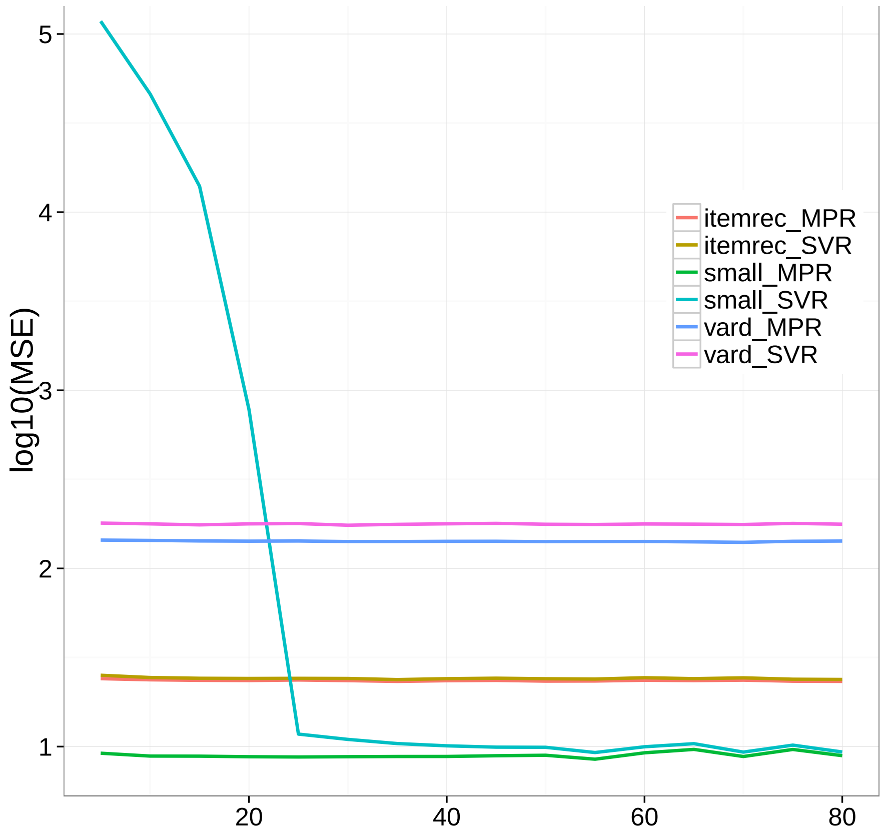


**Figure 5.5:** Plot of actual vs predicted values for three representative applications using two machine learning methods: SVR and polynomial regression.

In this context, the term, "sufficient amount of data" is introduced, which represents the auxiliary data requirement for the target application to be used in the two-mode transfer learning scheme. So, extensive experimentation is done to assess the right amount of data to assess what constitutes "sufficient" in practice, that needs to be collected as profiling data for the target application. A "sufficient amount" is identified by observing model convergence according to the change in percentage contribution of training data. To illustrate this, a graph is plotted for observed MSE values against the percentage of data which is utilised at training time to train the fitted model. This graph is shown in figure 5.5, where the x-axis indicates the percentage of training data set used to train a model and the y-axis represents the MSE values. The different colour lines indicate the models generated using SVR and polynomial regression methods by utilising the datasets of three representative applications running on Amazon EC2 instances. The line curves indicate the convergence of SVR-based and polynomial regression-based models. Except one SVR-based model, all models are converging quickly and are producing a constant prediction accuracy. A slow convergence is observed in one of the plotted SVR-based models which requires minimum of 22% data to be used as a training set. Note that the 100% data represents the collected application profiling data over a period of seven days using representative set of applications.

*Analysis* phase takes care of collecting a sufficient amount of data. This phase performs the same way as explained in Section 3.4.1, but the given number of application runs to collect profiling data is different under transfer learning setting. The amount of data collection is reduced to 22% which involves executing an application for nearly 36 hours. Recall steps for application profiling, as explained in Section 3.4.1, profiling information is collected by deploying the application on different virtual machines of a cloud provider involving pipeline of processes starting from getting application vignette, creating a virtual machine, deploying the application, and system monitoring. This collected information is then transformed into a readable format (CSV) for the *Learning* phase.

### 5.4.2 Similarity Measure

A similarity measure is required to identify which of the source data (source application) will give the best performance on the target domain (target application) to learn the target task (prediction model). A similarity measurement approach is proposed in this thesis considering it an essential pre-requisite for applying the transfer learning scheme. In this research, the similarity between the source and target domain is measured using two methods: *1) Profile comparison*, and *2) Kolmogorov–Smirnov (KS) test*

*Profile comparison* compares the target application metrics with the source application metrics saved in the knowledgebase. A metric is comprised of the cloud portfolio and application vignette as mentioned in Chapter 4. This is simply a value comparison for each attribute of application vignette. This comparison tries to find similarities at an application or cloud deployment level. Very often, it is difficult to compare deployment settings and resource utilisation between the two applications, especially when the virtual machines belong to different cloud providers and vary in configuration standards. Moreover, a simple comparison of resource utilisation is not an easy task due to the multiple level fine grained information. However, statistically, it is viable to quantify the level of similarity.

```
Input: (i) Auxiliary data = $S_{aux}$
(ii) Knowledgebase data = $S_{kb_i}$, where $i = 1...n$ and $n$=total number of applications
Output: (i) $D_d$ and $D_f$, a similarity estimate for each $S_{kb_i}$
(ii) Tagged knowledgebase datasets = $S_{kb_i}.tagged$
initialization;
  • Let $A_{j1}, ..., A_{jq}, .., A_{jk}$ be the value in $S_{aux}$, where $A_{j1}..A_{jq}$ represents application architecture,
    and $A_{jq}..A_{jk}$ represents deployment details
  • Let $B_{l1}, ..., B_{lq}, .., B_{lm}$ be the value in $S_{kb_i}$, where $B_{l1}..B_{lq}$ represents application architecture,
    and $B_{lq}..B_{lm}$ represents deployment details

Function SimilarityMeasure ($S_{aux}$, $S_{kb_i}$)
Start SimilarityMeasure
foreach $S_{kb_i} \in \{S_{kb_1}, ..., S_{kb_n}\}$ do
    foreach $x \in \{A_{j1}, ..., A_{jk}\} \cup \{B_{l1}, ..., A_{lm}\}$ do
        for $Two.Sample.KS.Test(x_a, x_b)$ do
            Compute $p$-value $\rightarrow D_p$
            Compute $D'$-value $\rightarrow D_d$
            if $p\text{-value} > 0.05$ then
                $x.mark = "SAME"$
            else
                if $D' < 0.5$ then
                    $x.mark = "SAME"$
                else
                    $x.mark = "DIFFERENT"$
                end
            end
        end
    end
    $aggregate.x.mark$ for $\{A_{j1}..A_{jq}\}$ and $\{A_{jq}..A_{jk}\}$
    if value of aggregate.x.mark is "SAME" for all $\{A_{j1}..A_{jq}\}$ then
        $S_{kb_i}.tagged = "SIMILAR"$
    else
        if value of aggregate.x.mark is "SAME" for >half of $\{A_{j1}..A_{jq}\}$ then
            $S_{kb_i}.tagged = "PARTLY-SIMILAR"$
        else
            $S_{kb_i}.tagged = "NOT-SIMILAR"$
        end
    end
end
END SimilarityMeasure
```

**Algorithm 2:** Similarity Measure

**Table 5.1:** KS test results

| **Feature** | **A & B.** | | **B & C** | | **A & C** | |
|---|---|---|---|---|---|---|
| | **D-value** | **p-value** | **D-value** | **p-value** | **D-value** | **p-value** |
| vmtype | 0.4082 | 2.20E-16 | 0.3986 | 2.20E-16 | 0.662 | 2.56E-07 |
| vcpu | 0.4063 | 2.20E-16 | 0.431 | 2.20E-16 | 0.0662 | 2.56E-07 |
| ecu | 0.4063 | 2.20E-16 | 0.431 | 2.20E-16 | 0.0662 | 2.56E-07 |
| ram | 0.4063 | 2.20E-16 | 0.431 | 2.20E-16 | 0.0662 | 2.56E-07 |
| day | 0.0804 | 2.32E-16 | 0.0926 | 2.20E-16 | 0.173 | 2.56E-07 |
| sub-time | 0.4063 | 2.20E-16 | 0.431 | 2.20E-16 | 0.0662 | 2.56E-07 |
| Ex-time | 1 | 2.20E-16 | 1 | 2.20E-16 | 0.633 | 2.20E-16 |
| apptype | 0.4063 | 2.20E-16 | 0.431 | 2.20E-16 | 0.0662 | 2.56E-07 |
| multi-threading | 0.4063 | 2.20E-16 | 0.431 | 2.20E-16 | 0.0662 | 2.56E-07 |
| external file | 0 | 1 | 1 | 2.20E-16 | 1 | 2.20E-16 |
| load in mem | 1 | 2.20E-16 | 1 | 2.20E-16 | 0 | 1 |
| parallel | 0 | 1 | 1 | 2.20E-16 | 1 | 2.20E-16 |
| file size | 1 | 2.20E-16 | 1 | 2.20E-16 | 1 | 2.20E-16 |
| **Result** | **Partly-similar** | | **Not-similar** | | **Not-similar** | |

The *KS test* is a statistical way of comparing the probability distribution of two separate datasets. This is a non-parametric test of the equality of continuous, one dimensional probability distribution to compare a sample with a reference probability distribution. It also quantifies the distance between the empirical distribution function of the two samples. In other words, the two-sample KS test checks whether two data samples come from the same distribution. However, it does not specify what that common distribution is (e.g. whether it's normal or otherwise).

The KS test is applied considering following reasons. First, the similarity test can be performed without knowledge about the common distribution of the source and target domain data. Second, it is sensitive to distribution. Third, it works well even if we do not know the mathematical distribution of observed properties of the dataset. Fourth, the application of KS test has no restriction on the sample size. Lastly, the KS test can be applied without restriction to any scientific problem, as this has been widely used in the astronomy domain as well, to find similarities amongst galaxies.

In the KS test the null distribution is calculated under a null hypothesis which postulates that both samples are drawn from the same distribution. The null and alternate hypotheses are stated:

$$H_0[x = y], H_1[\text{x differs from y}] \tag{5.1}$$

Algorithm 2 compares the probability distribution of the target application's profiling data ($S_{aux}$) with the existing applications ($S_{kbi}$) and identifies similar distributional application(s) ($S_{kb_i}.tagged$) from the knowledgebase. The application(s) are tagged according to the calculated values for distributional difference ($D_d$) and feature difference ($D_f$). The $KS$/similarity test is applied on vector inputs ($A_{j1}, ..., A_{jk}$ & $B_{l1}, .., B_{lm}$ ) from both source and target domain that needs distributional comparison. Each vector input represents a single feature from the source and target domains. The test outputs a p-value ($D_p$) and a D-value ($D_d$) for two samples each from source ($S_{kbi}$) and target ($S_{aux}$) application where p-value quantifies the probability of two samples populated from same or different distribution and D-value represents the difference of empirical distribution functions of two samples. The sub-level similarity tagging is assigned to each pair of features based on the p and D values.

If the p-value rejects the similarity hypothesis, then the D-value is evaluated to get an idea about the probability of similarity. A value in the range of 0-0.5 is considered as a measure of corresponding sample similarity from 50% to 100%. If we mark p-values and D-values accordingly we can get aggregated values to decide for similarity. The closeness is tagged as one of these three categories: 1) Similar, 2) Partly-Similar, or 3) Not-Similar. Table 5.1 presents the outcome of a similarity analysis based on three real world applications. The first 7 features in the table represent cloud and deployment related information and the remaining features explain application architecture. The three applications are named anonymously as A, B and C. Each D-value and p-value is calculated for each feature vector from the source and target applications as explained in "Function SimilarityMeasure()". The similar features are marked as "*". An application with high number of similar feature will get higher rank at similarity.

The aggregated similar features for A and B is higher than A and C so for application A being a target application B would be the first choice to be used as source of knowledge. Moreover, application A and B are closer in their defined architecture and are tagged as *Partly-Similar* for each other, however application C is tagged as *Not-Similar*. For application C both the applications A and B are ranked the same to be used as the source, however both applications have almost no similarity at application architecture level and will be tagged as *Not-Similar* to each other.

In a nutshell, the comparison looks for a similar application that has been logged in the knowledgebase. This similarity measure identifies the closeness of a new application (target domain) with the existing application(s) (source domain) present in the knowledgebase. The closeness is tagged as one of these three categories: *1) Similar*, *2) Partly-Similar*, or *3) Not-Similar*.

The SimilarityMeasure() method uses a builtin R function **ks.test()**. This function requires following parameters in this form:

$> ks.test(x, y, alternative = c("two.sided", "less", "greater"),$

$exact = NULL, tol = 1e-8, simulate.p.value = FALSE, B = 200)$

where $x$ and $y$ are numeric vector of data values and represents each column of a transformed CSV file, as this function tries to find distributional similarity between multiple columns of source and target application. 'alternative' indicates the alternative hypothesis and can take any of these values such as 'two-sided', 'less' and 'greater'. In this research we calculated ks test value using alternate hypothesis as 'two.sided' which calculates if the true distribution of $x$ is equal to $y$. 'exact' indicates whether an exact p-value should be calculated or not. 'tol' represents an upper bound for possible rounding an error. 'simulate.p.value' represents the inclusion of a Monte Carlo simulation to check goodness of fit and 'B' indicates the number of times this simulation will run.

### 5.4.3 Base Learner

Base learners are the machine learning methods used in the two-mode transfer learning scheme. The functionality of these learners is to learn a prediction model (target task). For this study, MPR and SVR are two machine learning methods used as the base learners and are referred as "MPR-learner" and "SVR-learner", respectively, throughout the chapter. There are some potential reasons for the selection of SVR and MPR as learners. A usage of common machine learning algorithms under traditional and transfer learning gives a fair comparison to highlight any potential benefits of one approach over the other. Moreover, we already have observed in Chapter 4 the prospective benefit of using these learning algorithms. The models which are generated using the traditional learning approach can be used for assessing the model accuracy with the transfer learning approach. Therefore, in order to avoid any further complexity and to prove the generality of the proposed approach, the same set of learning algorithms are used for transfer learning. Another important factor is the extensive use of SVM to solve classification problems using the transfer learning approach, which strengthens our decision of exploiting the use of SVM for regression [88].

### 5.4.4 Two-mode Transfer Learning Scheme

Two-mode transfer learning scheme aims in efficiently generating a learning model by following the proposed method which is based on the principle of transfer learning technique. As the name explains, the Two-mode transfer learning scheme has two modes and each mode follows different approaches of transfer learning for knowledge transfer across different domains. The working for this scheme is described in Algorithm 3. The two modes of this scheme are: 1) Transfer-All Mode and, 2) Transfer-Model Mode.

1. **Transfer-All Mode**

   This mode includes three approaches to transfer knowledge from the source to the target domain.

   (a) Transferring knowledge of feature representation

   (b) Transferring knowledge of instances

   (c) Transferring knowledge of parameter

2. **Transfer-Model Mode**

   This mode includes two approaches to transfer knowledge from the source to the target domain.

   (a) Transferring knowledge of feature representation

   (b) Transferring knowledge of parameter

This mode works by activating one of these modes and activation is based on the inputs from similarity measure and base learner. Similarity measure results in the identification of similarity between the source and the target application at the feature space and the marginal distribution level.

```
Input: (i) Auxiliary data = $S_{aux}$
(ii-a) Tagged knowledgebase datasets = $S_{kb_i}.tagged$
(ii-b) $X_{sig}$ of $S_{kb_i}.tagged$
(ii-c) Learning Method = $M$, where $M$ can be SVR or MPR
(ii-d) PredictionFunction = $f(S_{kb_i})$
Output: (i) PredictionFunction = $f(S_{aux})$
Function TwoModeTransferLearning ($S_{aux}$, $S_{kb_i}.tagged$, $M$)
Start TwoModeTransferLearning
Sort $S_{kb_i}.tagged$ based on tagged value, SIMILAR applications comes first
foreach $S_{kb_i}.tagged$ do
    if $S_{kb_i}.tagged$ == "SIMILAR" || $S_{kb_i}.tagged$ == "PARTLY-SIMILAR" then
        Set $D_d$ = FALSE
        if $M$ == $SVR$ then
            Set BaseLearner=$M$
            CallFunction TrasnferAll($S_{aux}$, $S_{kb_i}.tagged$,BaseLearner, $D_d$)
        else
            if $M$ == $MPR$ then
                Set BaseLearner=$M$
                CallFunction TrasnferModel($S_{aux}$, $S_{kb_i}.tagged$, BaseLearner, $D_d$)
            else

            end
        end
    else
        if $S_{kb_i}.tagged$ == "NOT-SIMILAR" then
            Set $D_d$ = TRUE
            if $M$ == $SVR$ then
                Set BaseLearner=$M$
                CallFunction TrasnferModel($S_{aux}$, $S_{kb_i}.tagged$, BaseLearner, $D_d$)
            else
                if $M$ == $MPR$ then
                    Set BaseLearner=$M$
                    CallFunction TrasnferModel($S_{aux}$, $S_{kb_i}.tagged$, BaseLearner, $D_d$)
                end
            end
        end
    end
end
END TwoModeTransferLearning
```

**Algorithm 3:** Two-mode Transfer Learning Scheme

Now, we explain the approaches involved in the designed scheme to transfer knowledge.

1. **Transferring knowledge of feature representation**

   The feature space represent specific properties regarding application architecture, deployment configurations and execution details as explained in Section Chapter 3. If both the source and target domains have some similarity at the application or deployment level, then the chances for effective contribution of same feature space are high while generating a learning model for the target domain.

   (a) If the feature space is same or nearly same in both source and target domain. The 'significant' features are selected and transferred from the source domain to the target domain. The significant features are the predictors which are identified while generating the prediction model for the source domain.

   (b) If the marginal distribution differs in both source and target domains due to the difference of application architecture. The identification of similar features is required in order to reduce the domain difference.

   (c) If the feature space differs in both source and target domains due to the varying standards of IaaS offerings. The mapping of similar features is required which can be done manually using the shared knowledge of both domains or automatically using the results of KS test.

2. **Transferring knowledge of instances**

   The instance knowledge represents a sample set comprised of the selected feature space. If the source and target domains have some similarity then the instance knowledge transfer can positively contribute for the model generation of target domain.

   (a) Transfer the instance knowledge of the selected feature space. The feature space is identified and selected during the knowledge transfer of feature representation.

   (b) The instance knowledge is transferred in an incremental way in order to avoid any influential effect of the source data.

3. **Transferring knowledge of parameter**

   The parameter knowledge details about the mathematical formulation of estimation or prediction function. This module transfers the parameter knowledge with respect to the selected base learner.

(a) If the base learner is SVR, transfer the kernel function, tuning parameters learning rates, learning-cost function and model constants.

(b) If the base learner is polynomial regression, transfer the polynomial order specific to each predictor, coefficient values and interaction terms.

For the purpose of detailed evaluation at each stage, we manually added model details based on the similarity measure. For example, if the application A is closely similar to application B we copied complete model structure from its source script to make it easy for knowledge transfer purpose.A self-explanatory code snippet is shown in Figure 5.6 explaining different steps involved in the model generation process. This process starts with reading source and data files and applying pre-processing method. This code tries to train model with increasing percentage of source data and target data. This process will run for multiple iterations with varying percentage of the source and target data in order to find any influential effect. All the outputs in each of the iteration are logged into a data file which then is evaluated by a human expert to detect any discrepancies regarding model generation (not a mandatory step).

### 5.4.5 Model Training & Assessment

Model training and assessment are two of the stages similar in both traditional settings as well as transfer learning setting. Details pertaining to functionality and method details is already explained in Section 3.4.2. The only difference in these methods under transfer learning is the distributional difference between test and training data set. Training data set includes data from the source and target applications, however, test data set is only comprised of data from the target application. As a model assessment result if the MSE is not satisfactory the next similar application is fetched to be used for model generation. A simple method to check goodness of fit is a comparison of training and test MSE.

## 5.5 Evaluation of the Two-mode Transfer Learning Scheme

This section provides evaluation of the two-mode transfer learning scheme. The main objective of this evaluation is to assess the generality of the proposed approach with a belief that the proposed scheme is able to make use of the learned knowledge to enhance model generation efficiency in terms of cost and time. This will lead to the achievement of the high level objective of efficient decision making.

```
dataSource=read.csv('path-to-file.csv',header=T) # read source file
 dataSource=na.omit(dataSource) # remove null values

 dataTarget=read.csv('path-to-file.csv',header=T) # read target file
 dataTarget=na.omit(dataTarget) # remove null values

 split_source=xPercent # default percentage is 5

 repeat
 {
   percentage=floor((60*totaldatasize(dataSource))/100)
   split_target=xPercent # sefault percentage is 5

   repeat
   {

     for (j in 1:200){ # train model in 200 iterations

       train1=sample(totaldatasize(dataSource),percentage)
       data_trainSource=dataSource[train1,]
       data_testSource=dataSource[-train1,]

       percentage1=floor((60*totaldatasize(dataTarget))/100)
       train2=sample(totaldatasize(dataTarget),percentage1)
       data_trainTarget=dataTarget[train2,]
       data_testTarget=dataTarget[-train2,]

       combineTrainingData=rbind(data_trainSource,data_trainTarget)


       Model=svm(responseVariable~ram+vcpu+ecu+externalfile+predictor + ... + ...+ ...., data=combineTrainingData, scale=FALSE,
                 kernel="polynomial",degree=3, ....., ....., ....)

       MSETraining=mean((combineTrainingData$responseVariable-predict(Model,combineTrainingData))^2) # calculate MSE on training
                                                                                                                        data

       .....
       .....

       df=data.frame(...., ...., .... ,....) # put details in dataframe and write in a table
       write.table(df, file='file-to-path.csv',row.names=FALSE,sep=",",col.names=FALSE, append=TRUE)

     } # end of for loop j

     MSETest=mean((data_testTarget$ttime-predict(Model,data_testTarget))^2) # calculate MSE on test data

     compareMSE(MSETest, MSETraining)  # compare test and training MSE for assessment
     incrementsplit_target ()
     if (split_target>60) break
   }
   } # end of for loop
   incrementsplit_source()
   if (split_source>60) break
 }
```

**Figure 5.6:** A code snippet of transfer learning setting.

We start evaluating this approach with three evaluation strategies in order to cover a wider span of possibilities, which are:

1. **Cross-application**. Can transfer learning scheme be applied across different applications that require deployment decisions on the same target cloud provider?
2. **Cross-provider**. Can transfer learning scheme be applied to assist in the deployment decisions of an application across different target cloud providers?
3. **Cross-application & cross-provider**. Can transfer learning scheme be applied to assist different applications for deployment decisions across different target cloud providers?

The evaluation process aims to assess the two-mode transfer learning scheme subject to the following fine-grained objectives:

1. Feasibility of approach with above mentioned evaluation strategies.
2. Rationality of the proposed scheme.
3. Assessment of the applied scheme.
4. Accuracy of the generated models.

We evaluate under certain assumptions. First, a large amount of data is available for mentioned evaluation strategies. Second, the data is normally distributed and outliers are removed. Third, the data contains no null values and all the column values are according to the set standard such as conversion of text values to numerical and unit conversion of file size. Fourth, knowledgebase is accessible which contains previously logged applications data as well as model details. Lastly, all the application packages are installed.

### 5.5.1 Experimental Details

The assumptions lead to the requirement of a large amount of data sets for multiple applications along with the learning models to represent application behaviour on different deployment setups. A substantial amount of data has already been collected by extensive experimentation on Amazon EC2 as stated in Chapter 4. The learning models were also generated for three representative applications.

**Table 5.2:** The computational specification of GCE instances.

| Series | Node | vCPU | GCEU | RAM (GB) | Storage (GB) | Price ($/h) |
|---|---|---|---|---|---|---|
| Standard Type | n1-standard-1 | 1 | 2.75 | 3.75 | 16(32 in Beta) | 0.042 |
| | n1-standard-2 | 2 | 5.5 | 7.5 | 16(64 in Beta) | 0.084 |
| | n1-standard-4 | 4 | 11 | 15 | 16(64 in Beta) | 0.168 |
| High Memory | n1-highmem-2 | 2 | 5.5 | 13 | 16(64 in Beta) | 0.106 |
| High CPU | n1-highcpu-2 | 2 | 5.5 | 1.8 | 16(64 in Beta) | 0.064 |
| | n1-highcpu-4 | 4 | 11 | 3.6 | 16(64 in Beta) | 0.128 |
| | n1-highcpu-8 | 8 | 2.2 | 7.2 | 16(64 in Beta) | 0.256 |

To completely explore all evaluation methods, we extend our experiment with the inclusion of another public cloud/IaaS provider: Google Compute Engine (GCE).

A substantial amount of data was also generated for GCE by following the same principles used for data collection for Amazon EC2. The experiments are continuously repeated using the same representative set of applications, over a period of seven days with a delay of ten minutes in between each pair of runs. The Linux tools `vmstat`, `glances` and `sysstat` are used to continuously monitor resource utilisation. The motive behind this activity was to generate comparative results to evaluate our approach and to highlight the benefit of transfer learning scheme.

Similar to Amazon EC2, GCE provides a differentiated series of instance types, catering to different application needs (eg compute-intensive, memory intensive, I/O-intensive, and so on). Each series contains a number of instance types offering different setups of computational resources. We targeted the Standard Type series n1-standard-1, n1-standard-2 and n1-standard-4. In addition, we selected the High CPU series n1-highcpu-2, n1-highcpu-4 and n1-highcpu-8 as well as the High Memory series n1-highmem-2 in order to evaluate varying combinations of resource capacities over a relatively wide price range. Only on-demand instances are used for this experiment. These have no long-term commitments and are charged on a pay-as-you-go basis at a 10 min rate.

Google Compute Engine uses KVM as the hypervisor which is used to launch virtual machines based on the 64 bit x86 architecture. All instances used run 64-bit Ubuntu Linux of different capacities as shown in Table 5.2. Google compute engine unit (GCEU), which is pronounced as GQ, is an abstraction of compute resources. According to Google, 2.75 GCEUs represent the minimum power of one logical core. Some of the information, such as details of parallel workload on virtual machines, scheduling algorithms and how GCE virtual cores are pinned to physical cores is not provided by the public cloud providers. So the users of infrastructure as a service cannot perceive any collocation or interference effect on their running application.

GCE differs from the Amazon-EC2 in various aspects such as the pricing scheme, virtual machine configuration measurement units and compute units. Amazon charges on an hourly basis for a virtual machines; in contrast, Google charges a minimum of 10 minutes per virtual machine. Both providers have non-standard categories to offer the pool of virtual machine's computational power and units.

Amazon uses the term 'ECU' as a computation unit to express the CPU capabilities of its various compute offerings while Google has defined its own computational unit as 'GCEU'. The capacity unit for measuring the disk size, machine type memory, and network usage are calculated in gigabytes (GB) for each EC2 instance of Amazon. Contrary to that, GCE uses gibibyte (GiB) as a measuring units for describing configurations of the virtual machines. It is very hard to make a 1:1 comparison with such a vague and non-standard description about the computational units and varying standards. This creates a difference of feature space at domain level. The proposed approach deals with such differences at the feature space level by mapping of similar features.

### 5.5.2 Evaluation 1: Cross-application

First evaluation strategy ascertain whether the two-mode transfer learning scheme is able to satisfy the objective of enhancing the learning efficiency in a cross-application scenario where the target provider is same for different applications. This can be explained using an example scenario.

*A prediction model for the application B can be generated to predict its performance on cloud X using the learned knowledge of application A having a prediction model to predict performance on cloud X and vice versa.*

The data source of three representative applications running on EC2 is utilised for this evaluation. For each of the target application as stated in Table 5.3, a source application is listed along with the similarity outcome. According to the similarity measurement, VARD and Item-Recommender are considered two of the applications which are closely similar to each other. However, rest of the test cases indicate no similarity at application level, the similarity output can be seen in Table 5.1.

Based on the similarity outcome and the activated transfer learning scheme, as stated in column 4-5, a prediction model is generated using the activated base learner. In order to assess the model accuracy, the MSE value of the generated model (as an outcome of two-mode transfer learning scheme) is compared with MSE value of the base model, as listed in the last two columns of Table 5.3.

The similarity at MSE values endorse that a learning model can be generated using the transferred knowledge of the similar application. It also validates the feasibility of applied transfer learning scheme for the representative test cases.

Moreover, the similar MSE results of both models confirms the applicability of the knowledge transfer approaches used for the applied transfer learning scheme. Besides, it also justifies feasibility of SVR and MPR as base learners to generate a learning model under transfer learning scheme. A similar result is presented to show the positive contribution of instance knowledge transfer which is one of the knowledge transfer approaches in the *Transfer-All* scheme.

Figure 5.7 shows the effect of instance knowledge (source application) transfer for the model generation of the target application. The effect is measured using MSE values on the test data set. The test data set belongs to the target application only; in contrast, the training dataset is composed of the mix of source and target domain data. The horizontal axis represents the percentage contribution of the source instances for the model generation, while the vertical axis lists the mean-MSE values for the test data. The left-hand plot in Figure 5.7 shows the effect of instance knowledge transfer of the VARD (source data) to generate the learning model (target task) for the Item-Recommender application (target domain). The consistent MSE value with low MSE confirms the positive contribution of the source data for the model generation. The right-hand plot also confirms the above justification when the instances from the Item-Recommender (source domain) are transferred to the model generation for the VARD (target domain). These results re-confirm the efficiency of the proposed scheme and the accuracy of the SVR-learner.

Comparable results are seen when the source and target applications have no similarity at application architecture level, as stated in Table 5.3. These test cases follow the *Transer-Model* scheme with both SVR and MPR as base learners. The model assessment result confirms that the feature representation is fairly describing both applications even when they belong to different categories. This also proves the generality of our designed generalised learning algorithm. In addition, the transferred model parameter details are able to generate a learning model to capture a different application's behaviour.

Besides the validation of applied transfer learning scheme, the model accuracy confirms the reduction in model generation time and training cost as only the 'sufficient' data was required from the target application to train the generated model. Moreover, the lengthy effort of model generation is reduced due to the use of existing knowledge.

The overall efficiency is achieved by saving 60% of required cost and time. This significantly has reduced a single virtual machine usage for data collection for one application from 168 hours to 67 hours. Consequently, saving a cost of $92.332 out of $153.888 on eight virtual machines of Amazon-EC2.

The 100% success rate validates the hypothesis for the viability of our approach across different applications for the same cloud provider. Further to that, the similarity of MSE values also endorses the accuracy of applied SVR-learner for transfer learning scheme. It also confirms that the efficiency can be achieved in terms of time and cost by making use of learned knowledge.

**Table 5.3:** Cross-application: Evaluation results of the two-mode transfer learning scheme

| **Test Case** | **Target** | **Source** | **Similarity** | **Applied Scheme** | **Generated Model** | | **Base Model** |
|---|---|---|---|---|---|---|---|
| | | | | | Base Learner | ≈ MSE | MSE |
| 1 | Item-Rec.-EC2 | VARD-EC2 | Yes | Transfer-All | SVR | 24.366 | 23.995 |
| | | | Yes | Transfer-Model | MPR | 23.362 | 23.362 |
| | | smallpt-EC2 | No | Transfer-Model | MPR | 23.362 | 23.362 |
| | | | No | Transfer-Model | SVR | 23.995 | 23.995 |
| 2 | VARD-EC2 | Item-Rec.-EC2 | Yes | Transfer-All | SVR | 177.180 | 177.960 |
| | | | Yes | Transfer-Model | MPR | 141.741 | 141.741 |
| | | smallpt-EC2 | No | Transfer-Model | SVR | 177.960 | 177.960 |
| | | | No | Transfer-Model | MPR | 141.741 | 141.741 |
| 3 | smallpt-EC2 | VARD-EC2 | No | Transfer-Model | SVR | 9.746 | 9.746 |
| | | | No | Transfer-Model | MPR | 8.983 | 8.983 |
| | | Item-Rec.-EC2 | No | Transfer-Model | SVR | 9.746 | 9.746 |
| | | | No | Transfer-Model | MPR | 8.983 | 8.983 |

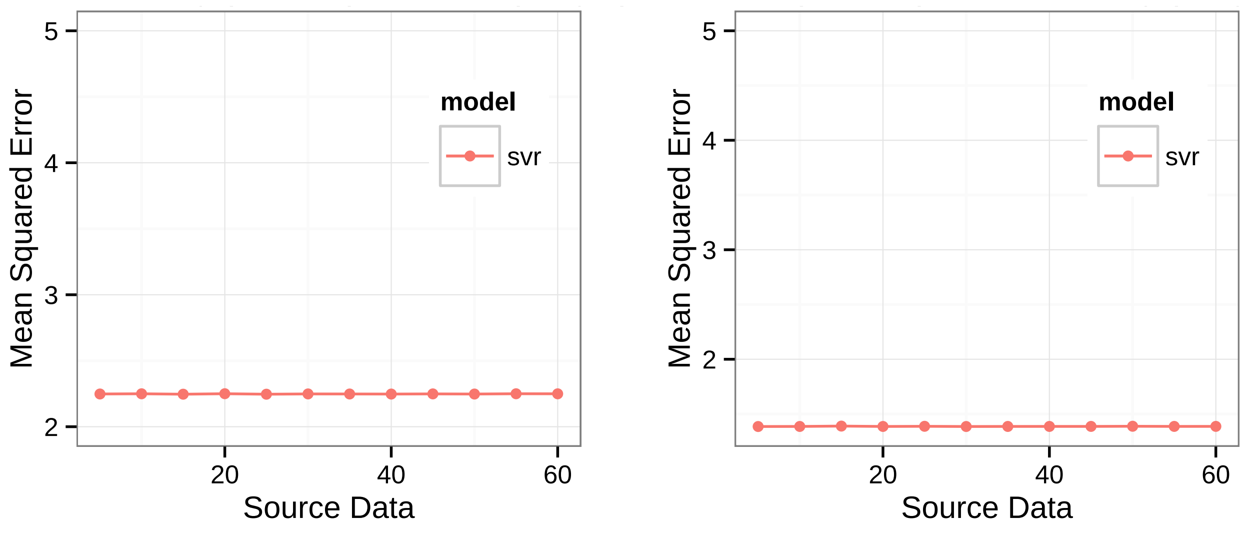


**Figure 5.7:** Left: Effect of instance knowledge transfer from the Item-Recommender to the VARD. Right: Effect of instance knowledge transfer from the VARD to the Item-Recommender.

### 5.5.3 Evaluation 2: Cross-provider

The second evaluation strategy ascertain whether the two-mode transfer learning scheme is able to satisfy the objective of enhancing the learning efficiency in a cross-provider scenario where the same application needs deployment decisions on different cloud providers. This can be explained using an example scenario.

*A prediction model for the application A can be generated to predict its performance on cloud Y using the learned knowledge of the same application having a prediction model to predict its performance on the cloud X and vice versa.*

This evaluation involves two cloud providers: Amazon EC2 and Google GCE. Therefore, the data sets collected from these two providers differ in feature space due to varying configuration standards at the virtual machine level, as stated earlier in this section. Table 5.4 lists all the test cases related to the described scenario. For each of the listed target application, a prediction model is generated using activated transfer learning scheme as well as the base learner.

Similar to the previous evaluation, the MSE values are assessed for the model accuracy. The model that is generated by following the transfer learning scheme is compared with the base model.

**Table 5.4:** Cross-provider: Evaluation results of the two-mode transfer learning scheme.

| **Test Case** | **Target** | **Source** | **Similarity** | **Applied Scheme** | **Generated Model** | | **Base Model** | **Comments** |
|---|---|---|---|---|---|---|---|---|
| | | | | | Base Learner | ≈ MSE | MSE | |
| 1 | VARD-GCE | VARD-EC2 | Yes | Transfer-All | SVR | 105.979 | 105.979 | |
| | | | Yes | Transfer-Model | MPR | 98.986 | 98.986 | |
| 2 | VARD-EC2 | VARD-GCE | Yes | Transfer-All | SVR | 162.7337- 177.639 | 177.960 | Reduced MSE |
| | | | Yes | Transfer-Model | MPR | 141.741 | 141.741 | |
| 3 | Smallpt-GCE | Smallpt-EC2 | Yes | Transfer-All | SVR | 0.341 | 0.341 | |
| | | | Yes | Transfer-Model | MPR | 0.3283 | 0.3283 | |
| 4 | Smallpt-EC2 | Smallpt-GCE | Yes | Transfer-All | SVR | 4.681-9.746 | 9.746 | Reduced MSE |
| | | | Yes | Transfer-Model | MPR | 8.983 | 8.983 | |
| 5 | Item-Rec.-EC2 | Item-Rec.-GCE | Yes | Transfer-All | SVR | 23.725 | 23.995 | |
| | | | Yes | Transfer-Model | MPR | 23.362 | 23.362 | |

Surprisingly, a reduction in MSE values is observed in two of the listed test cases with SVR as a base learner. In the first case, a prediction model is generated for the VARD-EC2 using the learned knowledge of VARD-GCE, listed under 2nd test case in Table 5.4. The observed MSE value 162.733 is significantly less than the base model MSE observed as 177.960. In the second case, the learned knowledge of smallpt-GCE is used to generate a prediction model for smallpt-EC2, as listed under 4th test case in Table 5.4. A reduction in MSE value is observed, where the base model value is 9.746 and the new model MSE is 4.681. This reduction at MSE level also indicates the valid source of information for the target domain, and we can say that VARD-GCE & smallpt-GCE are considered a good source of information for the two target domains such as VARD-EC2 & smallpt-EC2. This also confirms the positive influence of the transferred knowledge which validates the efficacy of applied transfer learning approach.

The results validates the feasibility of applied scheme and argues that it is possible to make use of the existing knowledge that can even help in increasing the model accuracy. The similar prediction results illustrate the generality of this approach in a cross-provider scenario as well.

This evaluation scenario also re-confirms the reduction in the model generation time and training cost as only a ‘sufficient’ amount of data is used from the target application for the training purpose.

Comparable results are seen when the proposed scheme is using SVR and MPR as base learners on the same test case, for example VARD-GCE and VARD-EC2 as target applications. The MPR learner is generating a model with reduced MSE compared to the SVR.

Another interesting fact is related to the capability of capturing data variation at model level. The source models are good in capturing the data variation of target domain in case of test cases on GCE, therefore, the model MSE values of all three applications are far less than the test cases running on EC2. This also indicates the high performance fluctuation at EC2 instances, as already described in Chapter 4.

A significant training overhead can be reduced up-to 67 hours out of 168 for a single virtual machine and more than 800 hours for a minimum of eight virtual machines. Hence, a cost of \$92.332 from total of \$153.888 can be saved on one single experiment comprised of 8 nodes on EC2. Similarly, on GCE, a cost of \$85.478 from total of \$142.464 can be saved on one single experiment comprised of 7 nodes. More precisely, an overall learning efficiency of 60% is achieved by saving on the actual cost and time.

### 5.5.4 Evaluation 3: Corss-application & Cross-provider

Third evaluation strategy ascertain, whether the two-mode transfer learning scheme satisfies the high-level objective of enhancing the learning efficiency, in a cross-application scenario when the target providers are different for both applications. Thus, providing deployment decisions across the different applications and cloud providers. An example scenario for such evaluation is stated as:

*A prediction model for the application B can be generated to predict its performance on the cloud Y using the learned knowledge of application A having a prediction model to predict performance on cloud X.*

**Table 5.5:** Cross-application & Cross-provider: Evaluation results of the two-mode transfer learning scheme.

| **Test Case** | **Target** | **Source** | **Similarity** | **Applied Scheme** | **Generated Model** | | **Base Model** |
|---|---|---|---|---|---|---|---|
| | | | | | Base Learner | ≈ MSE | MSE |
| 1 | VARD-GCE | Item-Rec.-EC2 | Yes | Transfer-All | SVR | 105.979 | 105.979 |
| | | Smallpt-EC2 | No | Transfer-Model | SVR | 105.979 | 105.979 |
| 2 | Item-Rec.-EC2 | VARD-GCE | Yes | Transfer-All | SVR | 23.725 | 23.995 |
| | | Smallpt-GCE | No | Transfer-Model | SVR | 23.995 | 23.995 |
| 3 | VARD-EC2 | Item-Rec.-GCE | Yes | Transfer-All | SVR | 177.6399 | 177.960 |
| | | Smallpt-GCE | No | Transfer-Model | SVR | 177.960 | 177.960 |
| 4 | Smallpt-EC2 | VARD-GCE | No | Transfer-Model | SVR | 9.746 | 9.746 |
| | | Item-Rec.-GCE | No | Transfer-Model | SVR | 9.746 | 9.746 |
| 5 | Smallpt-GCE | VARD-EC2 | No | Transfer-Model | SVR | 0.341 | 0.341 |
| | | Item-Rec.-EC2 | No | Transfer-Model | SVR | 0.341 | 0.341 |

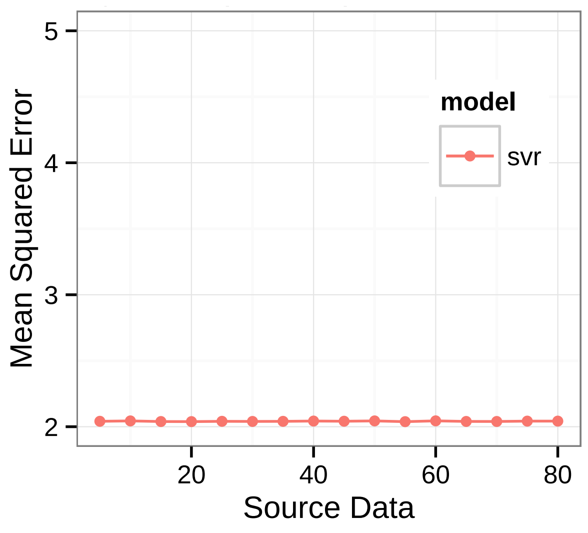
Similar to the previous two evaluations, the MSE value of the generated model is compared with the base model in order to assess the feasibility of two-mode transfer learning scheme. In this evaluation strategy, we are trying to conceive if the data of different applications with a considerable similarity at application architecture level can contribute towards a model generation for the target application.

For each of the target test case, list of the source applications are stated in Table 5.6 and 5.5. This evaluation considers test sets from different distribution and feature space, and assess if the applied scheme is able to extract and transfer useful data across different domains.

The similarity at the MSE values of all test cases confirms that the transfer learning scheme is then able to achieve the model generation using the knowledge of the source domain and source task, even when the domains have no similarity at the cloud or the application level. This also proves that a good feature representation can reduce the domain differences even if the domains have some heterogeneity at the cloud and application level. The model assessment results confirm that the feature representation is fairly describing both applications domain even when there is no similarity at application or cloud platform level. In addition, the transferred model parameters compliment with the feature representation and the generated model is able to capture different application's behaviour equally well.

Moreover, the efficiency is increased for the model generation by using the learned knowledge. In addition, the results validate the accuracy of the SVR-learner and MPR-learner used in a transfer learning technique.

A similar result is presented to show the positive contribution of instance knowledge transfer based on the application similarity even if the deployment setup differs in both. Figure 5.8 show the effect of using the instance knowledge for a given source domain (Item-Rec.-EC2) for generating a prediction model for the target domain (VARD-GCE) and vice versa. In addition, Figure 5.9 shows the positive influence of the source data (VARD-EC2) for a model generation for the target domain (Item-Rec.-GCE) and vice versa. The effect is measured using MSE values on the test data set. The test data set belongs to the target application only; in contrast, the training dataset is composed of the mix of source and target domain data. The horizontal axis represents the percentage contribution of the source instances for the model generation, while the vertical axis lists the mean-MSE values for the test data. In both figures, the consistent MSE value with low MSE confirms the positive influence of the source data for the model generation even when the applications belong to different deployment settings.



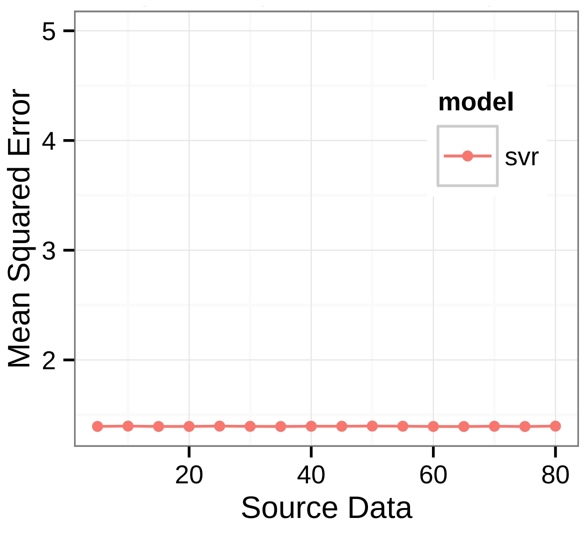
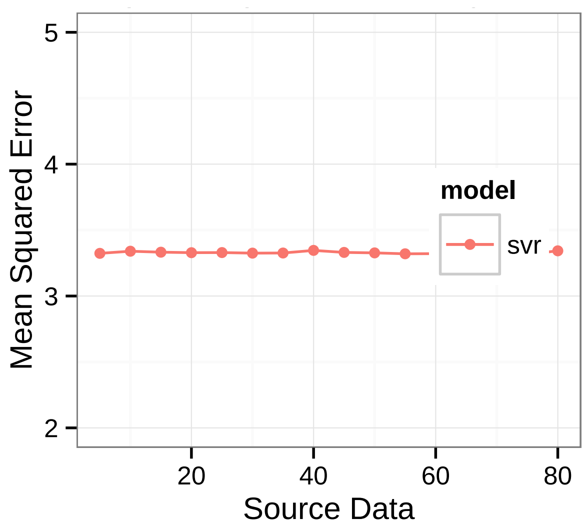


**Figure 5.8:** Left: Effect of Item-Rec.-EC2 instance knowledge (source domain) on the model generation for the VARD-GCE (Target domain). Right: Effect of VARD-GCE instance knowledge (source domain) on the model generation for the Item-Recommender-EC2 (Target domain).



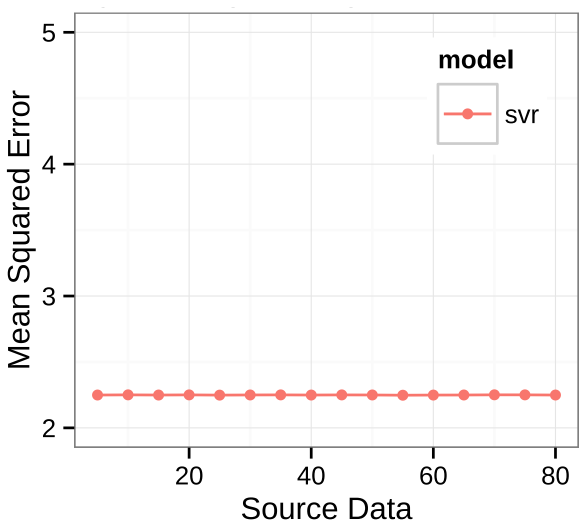


**Figure 5.9:** Left: Effect of VARD-EC2 instance knowledge (source domain) on the model generation for the Item-Recommender-GCE (Target domain). Right: Effect of movie-GCE instance knowledge (source domain) on the model generation for the VARD-EC2 (Target domain)

An overall learning efficiency of 60%, in terms of time and cost, is achieved by applying the transfer learning scheme. A significant training overhead can be reduced up-to 800 hours on 8 virtual machines. Hence a cost of more than $200 can be saved on both EC2 and GCE.

**Table 5.6:** Cross-application & Cross-provider: Evaluation results of the transfer learning scheme using polynomial regression method.

| **Test Case** | **Target** | **Source** | **Similarity** | **Applied Scheme** | **Generated Model** | | **Base Model** |
|---|---|---|---|---|---|---|---|
| | | | | | Base Learner | ≈ MSE | MSE |
| 6 | VARD-GCE | Item-Rec.-EC2 | Yes | Transfer-Model | MPR | 98.986 | 98.986 |
| | | Smallpt-EC2 | No | Transfer-Model | MPR | 98.986 | 98.986 |
| 7 | Item-Rec.-EC2 | VARD-GCE | Yes | Transfer-Model | MPR | 23.362 | 23.362 |
| | | Smallpt-GCE | No | Transfer-Model | MPR | 23.362 | 23.362 |
| 8 | VARD-EC2 | Item-Rec.-GCE | Yes | Transfer-Model | MPR | 141.741 | 141.741 |
| | | Smallpt-GCE | No | Transfer-Model | MPR | 141.741 | 141.741 |
| 9 | Smallpt-EC2 | VARD-GCE | No | Transfer-Model | MPR | 8.983 | 8.983 |
| | | Item-Rec.-GCE | No | Transfer-Model | MPR | 8.983 | 8.983 |
| 10 | Smallpt-GCE | VARD-EC2 | No | Transfer-Model | MPR | 0.328 | 0.328 |
| | | Item-Rec.-EC2 | No | Transfer-Model | MPR | 0.328 | 0.328 |

### 5.5.5 Limitation

There are three cases when the two-mode transfer learning scheme failed to generate an accurate learning model to predict the target application's performance. The Item-Recommender-GCE is the target application in all of these failing scenarios, as stated in Table 5.7.

We have observed a slightly different relationship of the response and one of predictor variable. This has given us motivation to further investigate towards this approach. This however, is part of our future work. An insightful idea is the exploration of relationship possibilities which leads towards using Bayesian approach.

**Table 5.7:** Failing Scenario. The two-mode transfer learning scheme could not help to generate a prediction model for the Item-Rec. (target domain) to predict performance on the GCE deployment setup.

| CS | Source Domain | Target Domain | Similarity | SVR MSE-SVR | MPR MSE-MPR | Base Model MSE |
|---|---|---|---|---|---|---|
| 1 | Item-Rec.-EC2 | Item-Rec.-GCE | Yes | 670.177 | 849.7076 | 243.445 |
| 2 | VARD-EC2 | Item-Rec.-GCE | Yes | 849.706 | 849.7076 | 243.445 |
| 3 | Smallpt-EC2 | Item-Rec.-GCE | No | 670.177 | 849.706 | 243.445 |

## 5.6 Summary and Discussion

The proposed approach leverages the use of a transfer learning technique and provides a scheme that can help in enhancing decision making efficiency by reducing the training overhead, thus achieving a learning model by making use of existing knowledge from a similar domain regardless of the change at data distributional level or feature space level. Hence, addressing two challenges related to learning cost and viability across application and cloud provider, as discussed in Section 4.5.2.

The proposed solution achieves the following. Firstly, the similarity measure method is able to identify relevancy at data distributional level in order to identify relevancy at domain level. Secondly, the two-mode scheme identifies the type of knowledge that can be used to achieve our goal. Thirdly, this scheme blocks transfer of knowledge that can have a negative impact on achieving the target task.

Experimental evaluation has proved that, by following this scheme, knowledge about the source application can be used for model generation of the target application. A model accuracy of more than 90% is observed for the generated models. Most importantly, this scheme fulfils the design objective of reducing the training overhead and improving the efficiency of model generation in terms of time and cost. Quantitatively, an overall reduction of 60% in terms of cost and time is observed. In some of the experiments, an increase in the model accuracy highlights that a better performing model can be achieved using transfer learning. The two-mode transfer learning scheme shows promising results to provide decisions across different applications and cloud platforms. Moreover, results also illustrate that the SVR and polynomial regression are complimentary to the adopted transfer learning techniques.

### 5.6.1 Learning Cost

One of the major challenges is related to learning cost that is involved in the traditional way of applying machine learning. It is evident that an initial cost is involved in the absence of any knowledge to create a knowledge base, but the presence of usable knowledge results in a huge reduction of learning cost. Let's evaluate the level of reduction in the light of experiments that we did the to generate learning models by following traditional machine learning approach as well as using transfer learning methodology.

The data collection cost has been reduced to 60% in the transfer learning approach. This significantly reduced a single virtual machine usage for data collection for one application from 168 hours to 67 hours. Consequently, saving a cost of \$92.332 out of \$153.888 on eight virtual machines of Amazon-EC2. Similarly, on GCE, a cost of \$85.478 from total of \$142.464 was saved on one single experiment comprised of 7 nodes. Same is the case with the number of days reduced from seven to less than three days. Now, consider learning time which has reduced from days to few seconds up to 3 minutes. This is a huge cut down in terms of learning time as there is no need to generate a new model from scratch involving evaluation of multiple stages and human expertise. The only implication, in this case, is presence and availability of knowledgebase. More precisely, an overall learning efficiency of 60% is achieved by saving on the actual cost and time.

The two-mode transfer learning scheme is a stepping stone towards developing an intelligent and efficient decision support system that can help generating decisions across different applications and cloud providers. We believe that a decision support system equipped with the novelty of machine learning and transfer learning is a cost-effective approach for the development of cloud brokers operating in a multi-cloud environment.

## 5.7 Potential Benefits of Intelligent Cloud Brokerage

We try to present end-end cost-benefit of Intelligent brokerage approach compared to a random choice of cloud configuration by considering an example demand: what would be cost and time of running 1000 jobs of x application? We can use the data collected during Analysis phase of this research work. We are selecting two applications of different architectures in relation to their intensity of memory (VARD) and CPU (smallpt) usage. Considering the observed variation on EC2 and GCE, we identify two cases which are labeled as Best Case and Worst Case. We now try to unravel cost-effectiveness and variation therein from the perspective of users who need to execute a certain number of jobs. We assume that each submitted job takes the same amount of time. The results are plotted in Figure 5.10, 5.11, 5.12, and 5.13 where we also indicate the amount of time needed for executing 1,000 jobs above each bar. With VARD on EC2, the cheapest instance `t2.small` is the most cost-effective: only \$0.75 to run 1,000. This cost rises between 6 and 8 folds for the most expensive instance. The amount of time reveals interesting facts as well. One can spend 27-29 hours to run 1,000 jobs on *t2.small* as opposed to 20-27 hours on *c4.xlarge* or 24-32 hours on `m3.large`. In effect, the user would pay much more cost for an uncertain reduction in execution time. *t2.medium* seems to be by far the most balanced in terms of cost and execution time: \$1.25 for 22-24 hours, almost 3-5 times cheaper than the expensive nodes and with a fairly certain and acceptable execution time.

For VARD on GCE, it is both time and cost effective to use the cheapest instance type `n1S1`, which can finish 1,000 job runs for just \$0.90 in exactly 25 hours. All other instance types are more expensive in time and cost. `c4.xlarge` provides the best cost:hour ratio compared to other EC2 instances, able to run 1,000 jobs for the same cost as with `c4.large`, but in nearly half the time. The same trend is noticed in `m3.xlarge` and `m3.large`.

Interestingly, the cost of smallpt jobs on GCE are almost equal on all instances except `n1S2` and `n1mem2`. In terms of time, `n1CPU8` (the most expensive per hour) takes only 58-59 hours which is less than half of the time needed on the next fastest instance type (`n1CPU4` and `n1S4`). This is a clear example illustrating that the cheapest instance is not necessarily the most cost- or time-effective.

smallpt also helps us draw a stark contrast between the two CSPs. Within the instance types we studied, GCE seems to outshine EC2 for executing smallpt jobs. Comparing `n1S2`, the second least cost- and time-effective GCE instance, to its EC2 counterparts: it is of equivalent performance and cost to `c4.large` but much cheaper than `m3.large`. Furthermore, general purpose GCE instance types extremely outperform the EC2 counterparts.

A cost and time are involved if there is no initial knowledge available with a broker. A possible solution to cut down cost and time is the availability of a range of real-world data traces and learning models for prediction. Supplementary model generation and initial data collection involve some cost and time, however, with the availability of initial data a broker is able to reduce this learning cost using transfer learning approach and so the cost will be cut down to 60% of the original. If we take the worst case example of spending 58-59 hours on a poorly performing node and recall the time required to collect auxiliary data which is 36 hours then spending these hours for analysis purpose and model generation can result in better performance and reduced cost. Now, if we assume that initial knowledge and auxiliary data both are available with a broker then the maximum learning time to generate a model is negligible compared to spending huge cost and time by choosing a worst-case option.

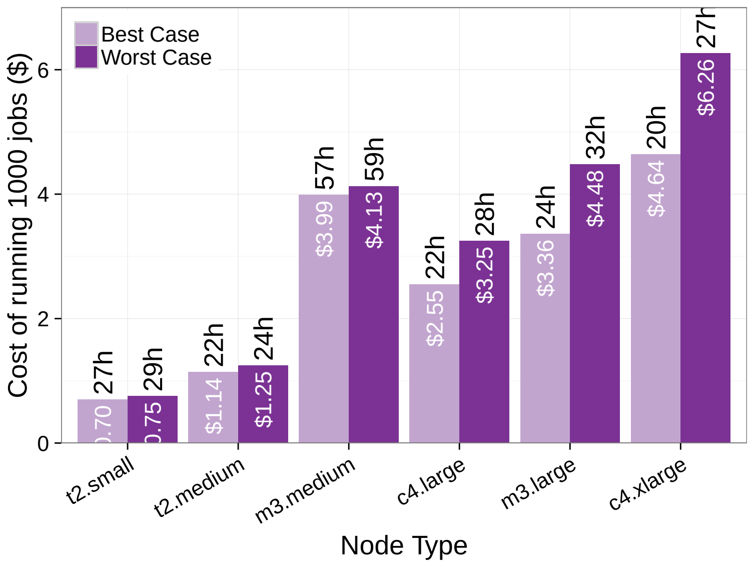


**Figure 5.10:** The cost of running 1000 VARD jobs on EC2.

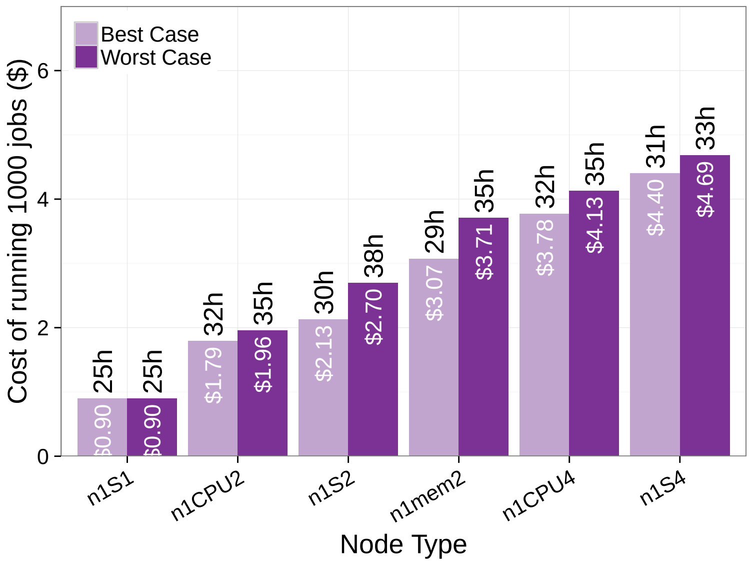


**Figure 5.11:** The cost of running 1000 VARD jobs on GCE.

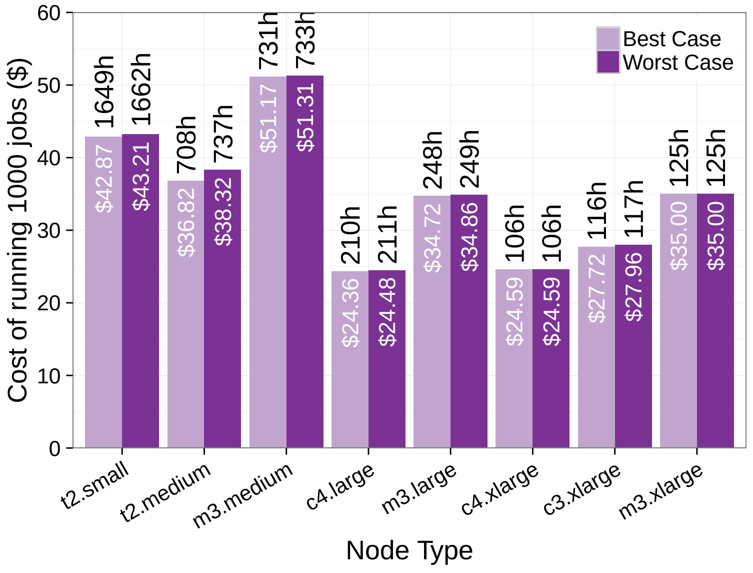


**Figure 5.12:** The cost of running 1000 smallpt jobs on EC2.

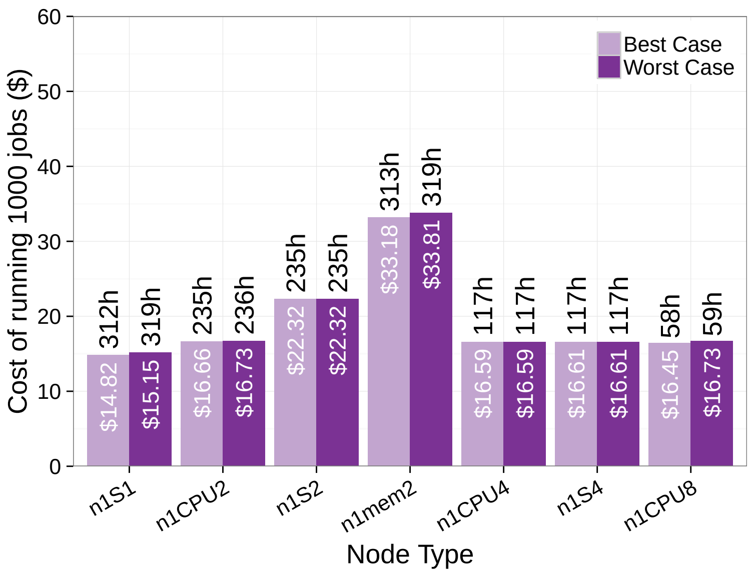


**Figure 5.13:** The cost of running 1000 smallpt jobs on GCE.

Chapter 6

# Conclusion

## 6.1 Introduction

The increase in magnitude and diversity in cloud service offerings at the IaaS layer has raised the complexity of decision making for cloud customers. Given this, it is increasingly important to offer decision support system as a fundamental component of a cloud brokerage architecture. Such a broker can assist a customer w.r.t application-specific requirements and customer-related constraints. Unfortunately, existing approaches are lacking in many ways: i) user friendliness, ii) providing application-driven and realistic solutions, and iii) flexibility in terms of dealing with different cloud providers and application domains.

In order to address this problem, this thesis puts forward an argument for developing an intelligent decision support system. The thesis investigates current solutions and forms a list of requirements deemed necessary to deliver an intelligent decision support solution. This decision support system makes use of machine learning techniques to provide behavioural and performance insight about the application and deployment setup necessary to make valid decisions.

Considering the fact that machine learning can impose significant training overhead, the efficiency of applying machine learning methods was also investigated. A key contribution of this thesis is the two-mode knowledge transfer scheme to make the intelligent decision support system more efficient across multi-cloud environments. This work adopts an experimental systems research methodology, with an iterative approach that is based on a quantitative analysis of real systems. This chapter provides an overall set of conclusions for the research, highlighting the main contributions and also listing possible areas of future work.

## 6.2 Thesis Summary

*Chapter 1* motivated and described the area of research, highlighting the main objective: Investigating the role of *machine learning* for designing a *decision support system* integrated with a *cloud broker* to assist customers in making application-driven deployment decisions across *multi-cloud* environments. This chapter also provided an introduction to the three broad domains underpinning the research, namely cloud brokerage, decision support systems and machine learning. Finally, the research methodology and the key contributions of the research work were also summarised.

*Chapter 2* surveyed the state of the arts in cloud brokers, decision support systems and machine learning, keeping in mind the core objective of the research work. First, the brokerage solutions were classified and presented in a three level taxonomic structure to give the reader a broad view of broker offerings and solutions. The three levels correspond to cloud management solutions, multi-cloud management services and goal optimisation. Second, a detailed analysis of related decision support systems was provided. Finally, the role of machine learning for decision support systems was discussed, highlighting the need for intelligent decision support system to provide realistic decisions.

*Chapter 3* described the architecture of Daleel. The fundamental part of Daleel is the intelligent decision support module enriched with different machine learning algorithms for the prediction of performance and assistance in IaaS selection. Finally, the selected machine learning methods were explained with their potential benefits.

*Chapter 4* presented an experimental evaluation of different learning strategies leading up to the adoption of a set of approaches. The experimental evaluation was performed using different instances of Amazon EC2 using a representative set of real-world applications. This chapter also explained how application-driven decisions can be taken using these learning models. The chapter also highlighting possible performance issues over training overhead. Chapter 4 concluded with the final architecture of a generic model predicting the performance of different applications.

*Chapter 5* investigated transfer learning techniques to enhanced the efficiency of an intelligent decision support system and reduce the problem of training overhead. In particular, the chapter introduced a novel two-mode transfer learning scheme with the goal of achieving substantial reduction in this overhead. The scheme was evaluated using two public cloud providers, i.e, Amazon Web Services (AWS) and Google cloud to validate the feasibility of proposed approach. Quantitatively, an overall training reduction of 60% was observed.

## 6.3 Contributions

The main contributions of this thesis are divided into two categories: the main overall contributions and other significant contributions. The main contributions coincide with the thesis objectives and the general contributions represent supporting knowledge that was achieved during this research work.

### 6.3.1 Main Contributions

1. **Architectural Insight**

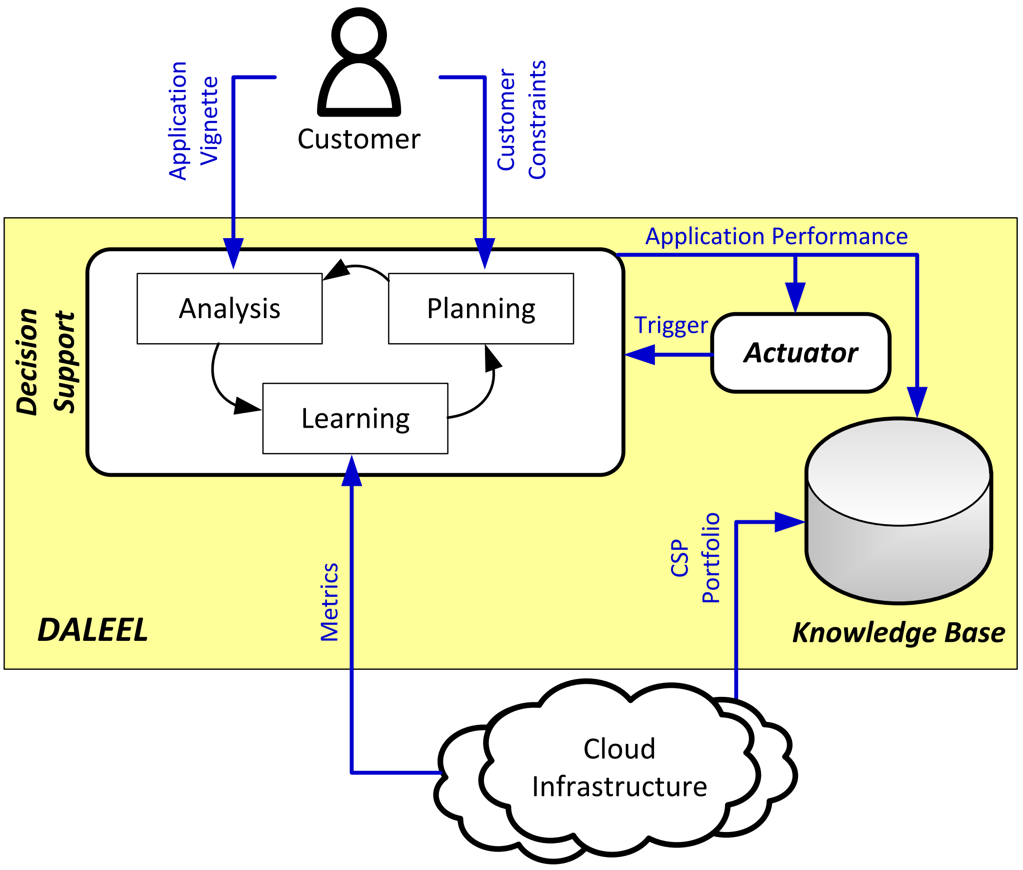


**Figure 6.1:** The Daleel Architecture.

An important contribution of this thesis is the architectural details of an intelligent decision support system, a key component of cloud brokerage. The complete architecture of Daleel, depicted in 6.1, consists of three primary architectural elements: *Decision Support*, *Actuator*, and *Knowledge Base*. The **Decision Support** module, equipped with machine learning models, is at the heart of Daleel's architecture and each of its module is responsible for performing tasks related to the learning objective. The framework adopts an iterative approach to incrementally determine the required machine learning methods. Moreover, the framework supports large-scale, back-end analysis to be fed iteratively into the model generation.

The initial design concept is published in the CrossCloud'14 workshop [5].

2. **Model Fitting Engine**

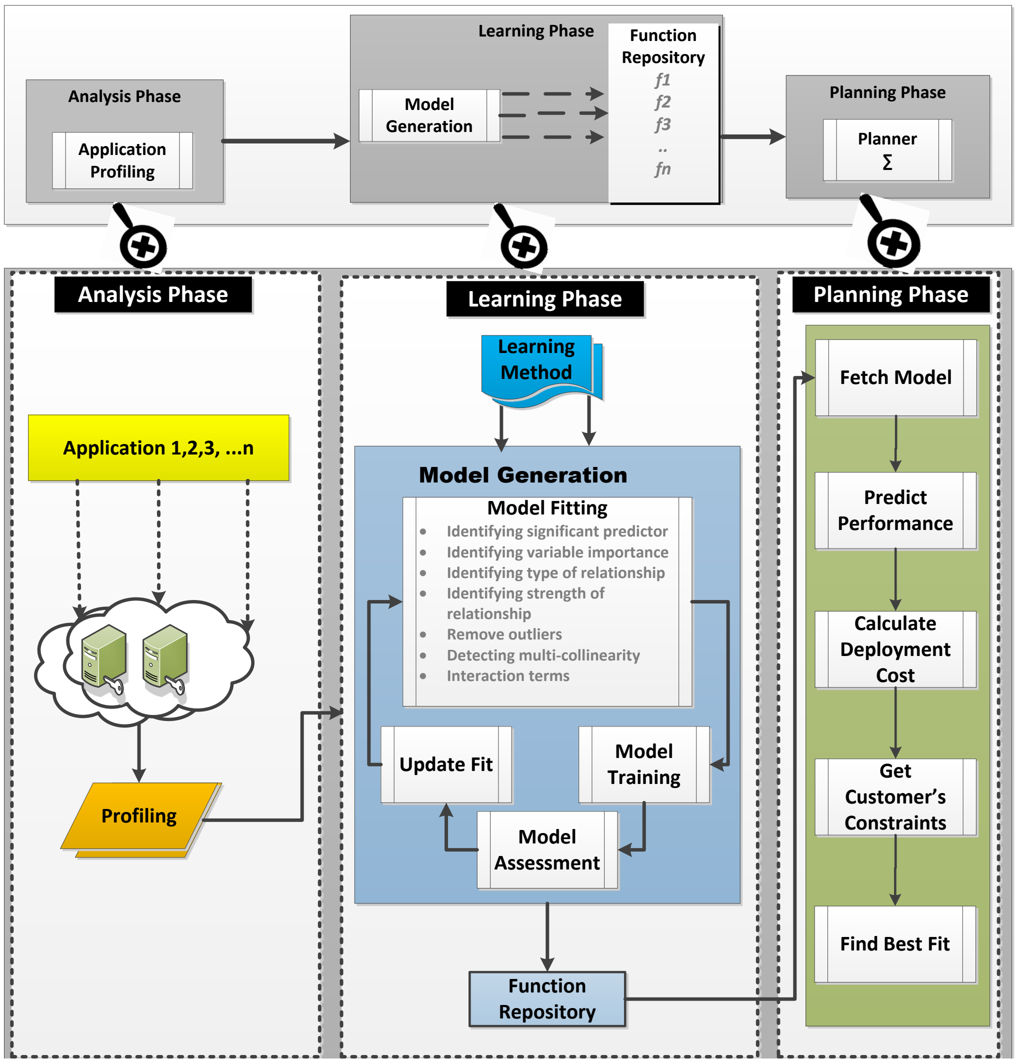


**Figure 6.2:** Model Fitting process

A second contribution of this thesis is the model fitting engine which is critical to this approach and provides support for decision making. Polynomial regression and SVR are two of the machine learning methods applied for the model generation. This engine is equipped with generalised as well as generic models to help with application-driven decisions. Some initial outcomes of this approach along with concrete architectural details are published in the NOMS conference 2016 [39].

3. **Two-mode transfer learning scheme**

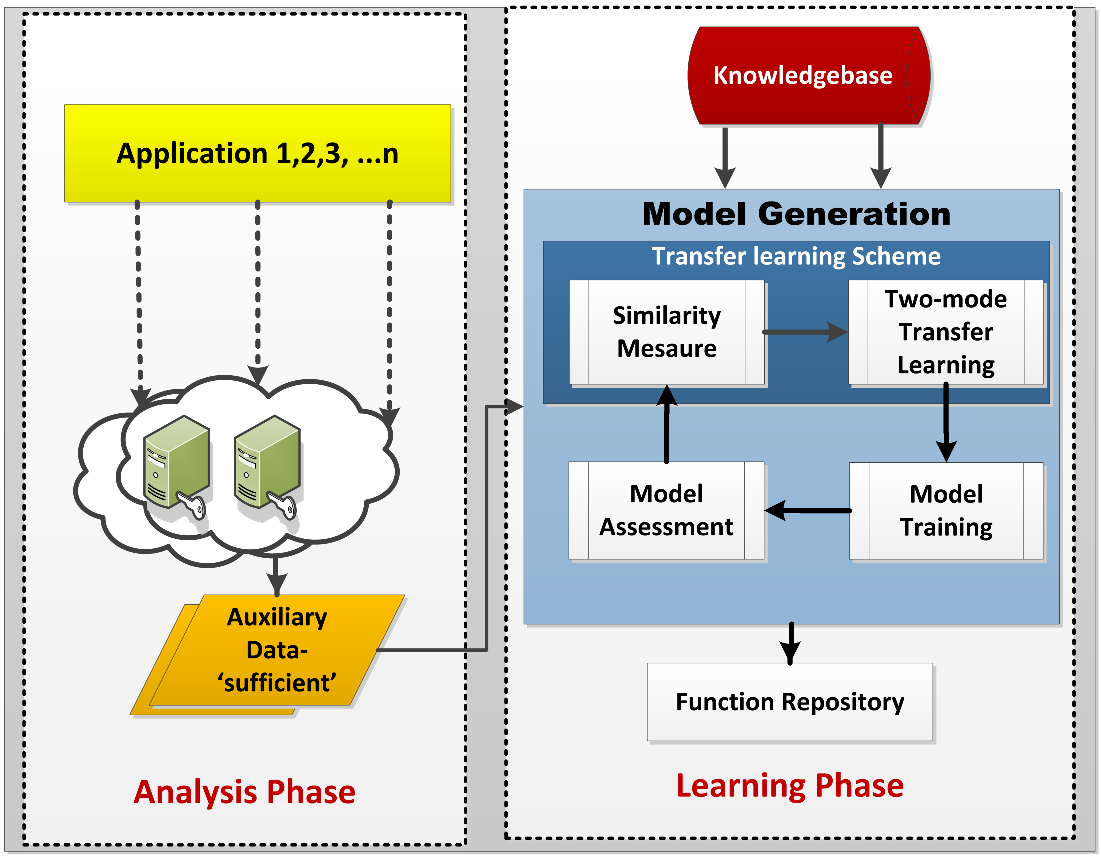


**Figure 6.3:** Two-mode transfer learning scheme.

The final major contribution of this thesis is the two-mode transfer learning scheme which is based on transferring knowledge from one domain to another using an approach based on semi-supervised learning, as shown in Figure 6.3. This technique significantly increase the efficiency of the intelligent decision support system by reducing the training overhead in terms of time and cost. Quantitatively, an overhead reduction of approximately 60% in the learning time and cost has been observed.

A manuscript is been drafted to be submitted to IEEE Transaction on Cloud Computing.

### 6.3.2 Other Significant Contributions

As well as the major contributions, a number of other interesting insights have emerged from the work:

1. The thesis includes a detailed study of machine learning techniques for decision support systems in general, and cloud brokerage in particular. This survey highlighted the huge diversity in machine learning techniques and the need to tailor solutions carefully for given problems, and even at a finer granularity, for sub-problems - like fitting a model for a particular application and for a given virtual machine.

2. The thesis also contains a comprehensive survey of cloud brokerage and related decision support system methodologies. This work provides an overall classification of existing approaches into cloud management, multi-cloud management, and goal optimisation. In addition to this, decision support systems were explored and classified according to applied methodologies.The survey highlighted that to date, there have been no successful integrated framework providing an end-to-end solution to assist customers with optimal deployment choice.

3. The empirical observation of application performance on instance types provided interesting insight into the *actual* performance of different virtual machine instances and how this varied from anticipated performance. This included some surprises. For example, in the experimental work carried out for the NOMS paper, it was identified that the M3 series of Amazon-EC2 is consistently under-performing. As of May 2017, Amazon has removed this series from their list of offerings. To someone unfamiliar with the IaaS market this might seem an insignificant change. However, based on the experience gained in this thesis, we have a clear explanation why , i.e. the virtual machine instance was not performing to specification.. This further illuminates the deep insight gained by the algorithmic machine learning approach presented in this thesis into the internals of IaaS operation.

## 6.4 Future Work

Working on this thesis has opened up a number of avenues for future research. In particular it would be interest to:

1. Enrich the algorithmic framework with supplementary learning methods and explore additional machine learning techniques to be used as base-learners for the semi-supervised and unsupervised transfer learning techniques such as Bayesian methods and Reinforcement learning.
2. Investigate classification methods to identify patterns and interesting clusters regarding different classes of application in different virtual machine settings.
3. Extend the study to deal with multi-criteria decision making to deal with trade-offs across multiple QoS attributes.
4. Extend the experimental evaluation using other categories of application and cloud providers to increase the understanding of the generality of the proposed approach.
5. More generally, it would be interesting to develop a full cloud broker architecture offering full independence from underlying cloud providers and also supporting a range of management options including migration and cloud bursting.

## 6.5 Revisiting the Research Goals

The main contributions of the thesis are reviewed by revisiting the research goals set in Section 1.5.

1. *The designing of a cloud broker architecture integrated with an implementation of an intelligent decision support system.*

   The first goal has been achieved by designing an integrated decision support architecture as discussed in Chapter 3.

2. *The investigation of machine learning methods that can be applied for optimal decision making in the decision support system of a cloud broker.*

   The second goal has been delivered by the model fitting engine which is populated by the learning approaches and generated models as detailed in Chapter 4.

3. *The development and evaluation of an efficient decision-making method integrated with the established decision support system to reduce the learning and decision-making cost.*

   The third goal has been accomplished by the two-mode transfer learning scheme as explained in Chapter 5.

# Appendix A

# Statistical Methods

## A.1 F-Statistics

F-statistics is a statistical test to measure the correlation between predictor and response with a given hypothesis. This test can be used to determine if we should keep the null hypothesis or not. F-statistics can be calculated using the formula as stated in equation A.1

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)} \tag{A.1}$$

F-statistic closer to 1 validates the absence of any relationship between X and Y and negates the hypothesis H1. On the other hand, F-statistic greater than 1 confirms the validity of hypothesis H1. The lower p-value corresponding to the F-statistic indicates the clear evidence of a relationship between response and predictor.

## A.2 Residual Sum of Squares (RSS)

RSS stands for residual sum of squares and can be defined as

$$RSS = {e_1}^2 + {e_2}^2 + .. + {e_n}^2 \tag{A.2}$$

$e_i = y_i - {y_i}'$ represents the ith *residual* which indicates the difference between the ith observed and ith predicted value. Here TSS represents the *total sum of squares* and measures the total variance in the response Y. TSS can be calculated as stated in equation A.3

$$TSS = \sum (y_i - y)^2 \tag{A.3}$$

## A.3 P-value

P-value is another statistical method to test statistical hypothesis. P-value defines the probability for a given statistical model when the null hypothesis is true. The lower the P-value the higher the chance to reject null hypothesis. If p-value is ¡0.5 then we reject the null hypothesis.

## A.4 R-squared

The $R^2$-value, is a statistical method for measuring the closeness of the actual and predicted data in terms of how similar the actual data are to the model fitted line or curve. The model fitted line or curve represents an estimated function (prediction function) for the actual data. $R^2$ can be calculated using following equation

$$R^2 = \frac{TSS - RSS}{TSS} \tag{A.4}$$

An $R^2$ value closer to 1 indicates that the regression line/curve explains the larger proportion of the variability in the response. In contrast, an $R^2$ value closer to 0 indicates that much of the variability is not explained by the regression. An adjusted R2 value along with different residual plots are also used for model assessment to evaluate the selection of significant predictors.

## A.5 Residual Standard Error (RSE)

RSE estimates the standard deviation of the response from the regression line. The $R^2$ statistic indicates the percentage of the variability recorded in the response that is explained by the predictors. The RSE measure can be calculated using following the formula given in Eq A.5

$$RSE = \sqrt{\frac{1}{n-2} RSS} \tag{A.5}$$

## A.6 Cross Validation (CV)

Cross validation is one of the widely used resampling methods for model selection. We used the k-fold cross validation method, computed by averaging the Mean Squared Error (MSE) for k-folds over the test sample using the following formula given in Eq:

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i$$

where $k = 20$ in our case. The MSE serves as a risk function for an estimator to measure the average of the squares of the error that is basically the difference between the estimator and estimated value [79]. It is calculated using following formula:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

## A.7 Variance Inflation Factor (VIF)

Collinearity can complicate or prevent the identification of an optimal set of explanatory variables for a statistical model. Collinearity can be identifies by observing variance inflation factors (VIF). VIF calculations are straightforward and easily comprehensible; the higher the value, the higher the collinearity. A VIF for a single predictor (explanatory variable) is obtained using the r-squared value of the regression of that variable against all other explanatory variables:

$VIF_i = \frac{1}{1-R_i^2}$

where the VIF for variable i is the reciprocal of the inverse of $R^2$ from the regression. A VIF is calculated for each predictor and those with high values are removed.

# Appendix B

# R Markdown

# Model fit

## Relationship between response and predictors

our first goal is to determine whether the provided data has some association among the response and predictor variables. This leads us to argue with our null hypothesis that shows the evidence of no relationship in response and predictor. If we find an evidence to reject the null hypothesis then the second step is to check the strength of that relationship, this is an important factor to express the level of accuracy for the prediction. Based on this relationship we can highlight those predictors having significant contribution towards prediction of response variable.Nature of this relation can be linear or non-linear and so we can start our experiment with regression.

In order to deal with the relationship exploration we have to adopt statistical ways to find out the answer. As a first step we apply linear regression and evaluate regression coefficients to accept or reject the null hypothesis. Coefficients with a zero value provides an evidence for the null hypothesis, this can be interpreted as no relationship among the response and predictors. Statistically we not only check coefficient values but adjusted R2 and P-value also. Adjusted R2 and p-value are some of the statistical ways to determine the association. R2, also known as coefficient of determination, is a statistical measure to show how close the data is with regression line. R2 close to 0 indicates that the model does not explain the variability in the response and as R2 close to 1 shows the model accuracy in terms of capturing the data variation in a given model.P-value less than 0.5 indicates the significance of the predictors within the model.

In our case,the predictors are mix of both hardware specific variables and application related variables. The hardware specific variables are related to ecu capacity, ram capacity and vcpu capaity. On the other hand, the application dependent variables indicate application type, threading information, external file requirement and load in memory information. We explore the relationship of response and predictors to see the association of application specific and hardware specific variables on response variable. ### Hardware specific predictors

```r
hfit=lm(ttime~ecu+vcpu+ram, data=data_train)
summary(hfit)
```

```
##
## Call:
## lm(formula = ttime ~ ecu + vcpu + ram, data = data_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -553.1 -269.9   -8.7  205.6 2034.3
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  790.104     16.614   47.56   <2e-16 ***
## ecu          103.169      4.019   25.67   <2e-16 ***
## vcpu        -643.518     20.250  -31.78   <2e-16 ***
## ram           60.060      2.392   25.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 458.6 on 6996 degrees of freedom
## Multiple R-squared:  0.143,  Adjusted R-squared:  0.1426
## F-statistic: 389.1 on 3 and 6996 DF,  p-value: < 2.2e-16
```

The significant predictors are marked as '*'under p-value column. A very less value of R2 shows that the multiple regression model, based on hardware specific predictors, is not showing much of the variability in the response, however p-value for all the predictors indicates their significance within model. ### Application specific predictors Multiple regression model based on application specific predictors, shows 36% of the variability in the response as indicated by R2 value. Most revealing thing in this model is the significance of the predictors which determines that only two variables, apptype and externalfile are contributing in this model and rest of the predictors are ignored which are indicated by NA under the p-value column. 'NA' more generally means that the coefficient is not estimable due to collinearity. Sometimes, it can also happen due to less observations to estimate the relevant parameters (e.g. if p>n). In our case the reason seems more collinearity rather than less observations.

```r
afit=lm(ttime~apptype+externalfile+loadinmem+threading, data=data_train)
summary(afit)
```

```
##
## Call:
## lm(formula = ttime ~ apptype + externalfile + loadinmem + threading,
##     data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -304.31 -234.33   -9.73   17.27 2032.17
##
## Coefficients: (2 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1381.39      50.03  27.613   <2e-16 ***
## apptype       -697.92      49.59 -14.074   <2e-16 ***
## externalfile   104.18      49.56   2.102   0.0356 *
## loadinmem          NA         NA      NA       NA
## threading          NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 396 on 6997 degrees of freedom
## Multiple R-squared:  0.3611, Adjusted R-squared:  0.361
## F-statistic:  1978 on 2 and 6997 DF,  p-value: < 2.2e-16
```

We have to look for those variables/predictors that can positively contribute towards explaining the variability of response in a model. Comparing all the models containing different subset of predictors is quite a hectic job, to deal with this task we look for best subset selection or variable selection as discussed in next section.

## Deciding on important variables

Variable selection determines that which of the predictors are associated with response to fit a model and are significant for deriving a robust model. Variable selection could be done by comparing a lot of models, each containing a different predictor subset. In order to find out the best model from a set of models, we use statistical method adjusted R2 and select the one that has highest adjusted R2 value. This is not always true that a model with high adjusted R2 value is the best or robust model so we do a step further to evaluate the model accuracy using cross validation, will be discussed in the section ahead. Forward selection, backward selection and mixed selection are methods for best subset selections and the quantifying measure is RSS. Below output is indicating the best possible combinations of predictors that can lead to a robust model in a linear, non-linear or with some interaction terms that we have explored in the last section.

```r
library(leaps)
leaps=regsubsets(ttime~. , data=data_train, nbest=7)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 2 linear dependencies found
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : nvmax reduced to 5
```

```r
summary(leaps)
```

```
## Subset selection object
## Call: regsubsets.formula(ttime ~ ., data = data_train, nbest = 7)
## 7 Variables  (and intercept)
##              Forced in Forced out
## ecu              FALSE      FALSE
## vcpu             FALSE      FALSE
## ram              FALSE      FALSE
## apptype          FALSE      FALSE
## externalfile     FALSE      FALSE
## threading        FALSE      FALSE
## loadinmem        FALSE      FALSE
## 7 subsets of each size up to 5
## Selection Algorithm: exhaustive
##          ecu vcpu ram apptype externalfile threading loadinmem
## 1  ( 1 ) " " " "  " " " "     " "          " "       "*"
## 1  ( 2 ) " " " "  " " "*"     " "          " "       " "
## 1  ( 3 ) " " " "  " " " "     " "          "*"       " "
## 1  ( 4 ) " " " "  " " " "     "*"          " "       " "
## 1  ( 5 ) " " "*"  " " " "     " "          " "       " "
## 1  ( 6 ) "*" " "  " " " "     " "          " "       " "
## 1  ( 7 ) " " " "  "*" " "     " "          " "       " "
## 2  ( 1 ) " " "*"  " " " "     " "          " "       "*"
## 2  ( 2 ) " " "*"  " " "*"     " "          " "       " "
## 2  ( 3 ) " " "*"  " " " "     " "          "*"       " "
## 2  ( 4 ) " " "*"  " " " "     "*"          " "       " "
## 2  ( 5 ) "*" " "  " " "*"     " "          " "       " "
## 2  ( 6 ) "*" " "  " " " "     " "          " "       "*"
## 2  ( 7 ) "*" " "  " " " "     " "          "*"       " "
## 3  ( 1 ) "*" "*"  " " " "     " "          " "       "*"
## 3  ( 2 ) "*" "*"  " " "*"     " "          " "       " "
## 3  ( 3 ) " " "*"  "*" " "     " "          " "       "*"
## 3  ( 4 ) " " "*"  "*" "*"     " "          " "       " "
## 3  ( 5 ) " " "*"  " " " "     " "          "*"       "*"
## 3  ( 6 ) " " "*"  " " " "     "*"          " "       "*"
## 3  ( 7 ) " " "*"  " " "*"     " "          "*"       " "
## 4  ( 1 ) "*" "*"  "*" " "     " "          " "       "*"
## 4  ( 2 ) "*" "*"  "*" "*"     " "          " "       " "
## 4  ( 3 ) "*" "*"  " " "*"     " "          "*"       " "
## 4  ( 4 ) "*" "*"  " " "*"     "*"          " "       " "
## 4  ( 5 ) "*" "*"  " " " "     "*"          " "       "*"
## 4  ( 6 ) "*" "*"  " " " "     " "          "*"       "*"
## 4  ( 7 ) "*" "*"  " " "*"     " "          " "       "*"
## 5  ( 1 ) "*" "*"  "*" " "     " "          "*"       "*"
## 5  ( 2 ) "*" "*"  "*" " "     "*"          " "       "*"
## 5  ( 3 ) "*" "*"  "*" "*"     "*"          " "       " "
## 5  ( 4 ) "*" "*"  "*" "*"     " "          "*"       " "
## 5  ( 5 ) "*" "*"  "*" "*"     "*"          " "       " "
## 5  ( 6 ) "*" "*"  " " "*"     "*"          " "       "*"
## 5  ( 7 ) "*" "*"  " " " "     "*"          "*"       "*"
```

7 subsets of each size up to maximum significant predictors are shown in the above output. In our case a limit of 5 predictors is indicated as significant in the respective models with variable set of predictors. we can also use forward selection or backward selection to get best subset of predictors.

```r
regft.fw=regsubsets(ttime~., data=data_train, nvmax=5,method="forward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 2 linear dependencies found
```

```r
summary(regft.fw)
```

```
## Subset selection object
## Call: regsubsets.formula(ttime ~ ., data = data_train, nvmax = 5, method = "forward")
## 7 Variables  (and intercept)
##              Forced in Forced out
## ecu              FALSE      FALSE
## vcpu             FALSE      FALSE
## ram              FALSE      FALSE
## apptype          FALSE      FALSE
## externalfile     FALSE      FALSE
## threading        FALSE      FALSE
## loadinmem        FALSE      FALSE
## 1 subsets of each size up to 5
## Selection Algorithm: forward
##          ecu vcpu ram apptype externalfile threading loadinmem
## 1  ( 1 ) " " " "  " " "*"     " "          " "       " "
## 2  ( 1 ) " " "*"  " " "*"     " "          " "       " "
## 3  ( 1 ) "*" "*"  " " "*"     " "          " "       " "
## 4  ( 1 ) "*" "*"  "*" "*"     " "          " "       " "
## 5  ( 1 ) "*" "*"  "*" "*"     "*"          " "       " "
```

```r
regft.bw=regsubsets(ttime~., data=data_train, nvmax=5,method="backward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 2 linear dependencies found
```

```r
summary(regft.bw)
```

```
## Subset selection object
## Call: regsubsets.formula(ttime ~ ., data = data_train, nvmax = 5, method = "backward")
## 7 Variables  (and intercept)
##              Forced in Forced out
## ecu              FALSE      FALSE
## vcpu             FALSE      FALSE
## ram              FALSE      FALSE
## apptype          FALSE      FALSE
## externalfile     FALSE      FALSE
## threading        FALSE      FALSE
## loadinmem        FALSE      FALSE
## 1 subsets of each size up to 5
## Selection Algorithm: backward
##          ecu vcpu ram apptype externalfile threading loadinmem
## 1  ( 1 ) " " " "  " " "*"     " "          " "       " "
## 2  ( 1 ) " " "*"  " " "*"     " "          " "       " "
## 3  ( 1 ) "*" "*"  " " "*"     " "          " "       " "
## 4  ( 1 ) "*" "*"  "*" "*"     " "          " "       " "
## 5  ( 1 ) "*" "*"  "*" "*"     "*"          " "       " "
```

The following subset of predictors are selected from the above outcome and nearly all the subsets shows that hardware specific variables are considered more significant. The models are divided into 3 categories, the first category contains the 4 variable based models where all of the hardware predictors are used with inclusion of just one application specific variable. The second category is composed of 5 variable model which include all hardware specific variables and 2 of the application specific variables. The third category has the models in which one of the hardware specific variable “ram”" is excluded and 3 of the application specific variables are used in each model.

4 variable with inclusion of all hardware specific predictors ecu+vcpu+ram+apptype ecu+vcpu+ram+loadinmem ecu+vcpu+ram+externalfile ecu+vcpu+ram+threading

5 variable with inclusion of two application specific predictors ecu+vcpu+ram+threading+loadinmem ecu+vcpu+ram+externalfile+loadinmem ecu+vcpu+ram+apptype+externalfile ecu+vcpu+ram+apptype+threading

5 variable with exclusion of one hardware predictor ecu+vcpu+apptype+externalfile+loadinmem ecu+vcpu+externalfile+threading+loadinmem

## Model fit

We will use these selected subset predictors in a rgression model and see the output. ### 4 variable–all hardware + 1 application specific

The models comprised of all hardware specific variables and one application specific variable shows large proportion of the data variability, however inclusion of either apptype or loadinmem shows more than 61% of response variability that is higher than other models that are using rest of the application specific variables. To check the percentage of response variability, have a look at the adjusted R2 value of each model.

```
fit1=lm(ttime~ecu+vcpu+ram+apptype, data=data_train)
summary(fit1)
```

```
##
## Call:
## lm(formula = ttime ~ ecu + vcpu + ram + apptype, data = data_train)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -369.43 -144.57  -98.06    2.62 1479.33
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2301.675     19.815 116.157   <2e-16 ***
## ecu           53.964      2.751  19.618   <2e-16 ***
## vcpu        -490.814     13.698 -35.830   <2e-16 ***
## ram           15.915      1.676   9.497   <2e-16 ***
## apptype     -796.127      8.625 -92.302   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 308 on 6995 degrees of freedom
## Multiple R-squared:  0.6136, Adjusted R-squared:  0.6134
## F-statistic:  2777 on 4 and 6995 DF,  p-value: < 2.2e-16
```

```
fit2=lm(ttime~ecu+vcpu+ram+loadinmem, data=data_train)
summary(fit2)
```

```
##
## Call:
## lm(formula = ttime ~ ecu + vcpu + ram + loadinmem, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -369.43 -144.57  -98.06    2.62 1479.33
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1505.548     13.585 110.828   <2e-16 ***
## ecu           53.964      2.751  19.618   <2e-16 ***
## vcpu        -490.814     13.698 -35.830   <2e-16 ***
## ram           15.915      1.676   9.497   <2e-16 ***
## loadinmem   -796.127      8.625 -92.302   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 308 on 6995 degrees of freedom
## Multiple R-squared:  0.6136, Adjusted R-squared:  0.6134
## F-statistic:  2777 on 4 and 6995 DF,  p-value: < 2.2e-16
```

```
fit1a=lm(ttime~ecu+vcpu+ram+externalfile, data=data_train)
summary(fit1a)
```

```
##
## Call:
## lm(formula = ttime ~ ecu + vcpu + ram + externalfile, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -252.98 -147.84  -97.30    6.67 1488.90
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  1490.671     14.039  106.18   <2e-16 ***
## ecu            57.042      2.833   20.13   <2e-16 ***
## vcpu         -502.994     14.119  -35.63   <2e-16 ***
## ram            18.117      1.725   10.50   <2e-16 ***
## externalfile -771.233      8.854  -87.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 317.7 on 6995 degrees of freedom
## Multiple R-squared:  0.5889, Adjusted R-squared:  0.5887 
## F-statistic:  2505 on 4 and 6995 DF,  p-value: < 2.2e-16
```

```r
fit1b=lm(ttime~ecu+vcpu+ram+threading, data=data_train)
summary(fit1b)
```

```
##
## Call:
## lm(formula = ttime ~ ecu + vcpu + ram + threading, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -252.98 -147.84  -97.30    6.67 1488.90 
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  -51.795     15.028  -3.447 0.000571 ***
## ecu           57.042      2.833  20.131  < 2e-16 ***
## vcpu        -502.994     14.119 -35.627  < 2e-16 ***
## ram           18.117      1.725  10.502  < 2e-16 ***
## threading    771.233      8.854  87.106  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 317.7 on 6995 degrees of freedom
## Multiple R-squared:  0.5889, Adjusted R-squared:  0.5887 
## F-statistic:  2505 on 4 and 6995 DF,  p-value: < 2.2e-16
```

#### 5 variable–all hardware specific predictors + 2 app specific

The five variable models are using all of the hardware and two of the application specific predictors. If we compare adjusted R2 value with above models, we find out the same value. However in each model one of the application specific variable is shown unsignificant that we can see under p-balue column. The possible reason could be collinearity among the variables and so further diagnostics will be used to evaluate it.

```r
fit3=lm(ttime~ecu+vcpu+ram+threading+loadinmem, data=data_train)
summary(fit3)
```

```
##
## Call:
```

```
## loadinmem    -842.617     38.923 -21.648   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 310 on 6995 degrees of freedom
## Multiple R-squared:  0.6086, Adjusted R-squared:  0.6084
## F-statistic:  2720 on 4 and 6995 DF,  p-value: < 2.2e-16
```

## Regularization

The subset selection method use least square to fit a linear model that comprised of a subset of predictors. In extension to our linear assumption We can use another technique that uses all the predictors in the model fit and can shrink some of the coefficeints estimates to zero. This shrinkage of coeffiecint estimates is also known as regularization. The two well known techniques for this method are the ridge regression and lasso. ### Ridge regression Ridge regression is similar to least squares but minimises the coefficient estimates with a slightly different quantity of lambda. Lambda is a tuning parameter that controls the relative impact of the least square and shrinkage penalty on the regression coefficient estimates. When lambda=0, the penalty term has no effect and estimates are least square. However, as lambda grows to infinity the shrinkage penalty grows and the coefficient estimates approaches zero.

Ridge regression includes all the variables as P predictors in the final model. Highest value of lambda can reduce the coefficient value but cannot exclude any variable from the resulting model. On the other hand, Lasso overcomes this disadvantage by forcing some of the coefficient estimates to be equal to zero especially when the lambda value is large enough. For the ridge and lasso regression model fit, we use a range of values starting from lamda= 10 power 10 to lambda=10 power -2, covering the full range of scenarios from the null model containing only the intercept to the least square fit.

```
library(glmnet)
```

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-2
```

```
  grid=10^seq(10,-2,length=100)
  x=model.matrix(ttime~.,data=data_train)[,-1]
  y=data_train$ttime
ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
dim(coef(ridge.mod))
```

```
## [1]   8 100
```

Output will be a 8x100 (8 predictors and 100 colum values for lambda) matrix which shows coefficinets values for each value of lambda, can be seen from the output of dim() function.

We expect the coefficeint estimates to be much smaller, in terms of l2 norm when a lareg value of lambda is used as compared to when a small value of lambda is used. check if coefficeinet estimate of lambda[50]= 11,498 is higher than lambda[60]=705

```
ridge.mod$lambda[50]
```

```
## [1] 11497.57
```

```
 coef(ridge.mod)[,50]
```

```
##  (Intercept)          ecu         vcpu          ram      apptype
## 437.54070780  -0.52608162  -3.51598432  -0.09832739 -22.19444952
## externalfile    threading    loadinmem
## -21.54179748  21.54179295 -22.19443095
```

```
sqrt(sum(coef(ridge.mod)[-1,50]^2))
```

```
## [1] 43.88545
```

```
ridge.mod$lambda[60]
```

```
## [1] 705.4802
```

```
coef(ridge.mod)[,60]
```

```
##  (Intercept)          ecu         vcpu          ram      apptype
##   784.250996    -7.819823   -51.736508    -5.217145  -134.125738
## externalfile    threading    loadinmem
##  -123.270345   123.276213  -134.109438
```

```
 sqrt(sum(coef(ridge.mod)[-1,60]^2))
```

```
## [1] 262.931
```

Change the parameter value as per predictor/variable values. We can use predict() func to obtain ridge regression coefficeints for a new value of lambda lets say 50.

```
 predict(ridge.mod,s=50,type="coefficients")[1:8,]
```

```
##  (Intercept)          ecu         vcpu          ram      apptype
##  1311.984584    -2.598538  -185.462669    -3.719652  -256.142263
## externalfile    threading    loadinmem
##  -135.136443   137.826054  -258.706751
```

Fit a ridge regression model on the trining set and evaluate its MSE on the test set using lambda=4.

```
train=sample(1:nrow(x),nrow(x)/2)
test=(-train)
y.test=y[test]
ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid,thresh=1e-12)
ridge.pred=predict(ridge.mod ,s=4, newx=x[test,])
mean(( ridge.pred -y.test)^2)
```

```
## [1] 89379.79
```

1e10 means 10 power 10, remember that least square is simply ridge regression with lambda=0.

```r
 ridge.pred=predict(ridge.mod ,s=1e10, newx=x[test,])
 mean(( ridge.pred -y.test)^2)
```

```
## [1] 228212.6
```

```r
 ridge.pred=predict(ridge.mod ,s=0, newx=x[test,], exact=T)
mean(( ridge.pred -y.test)^2)
```

```
## [1] 89213.3
```

The lm is more useful to see output if we want to fit unpenalized least square model.

```r
 lmridge=lm(y~x, subset=train)
summary(lmridge)
```

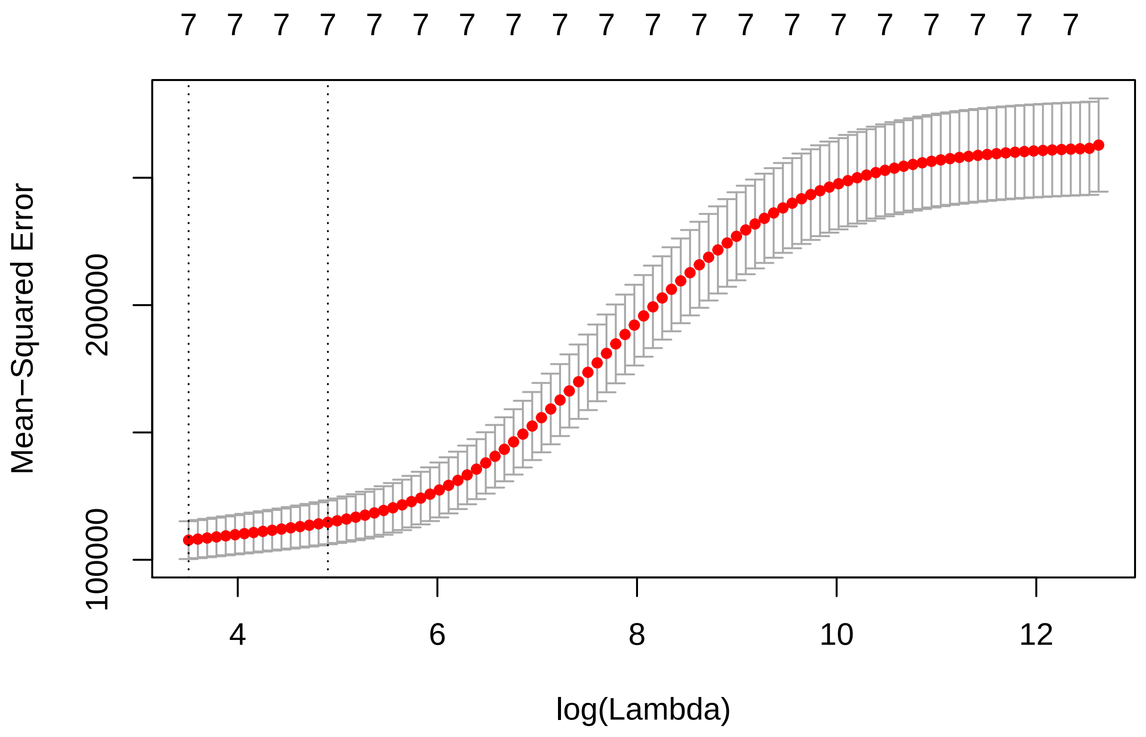
```
##
## Call:
## lm(formula = y ~ x, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -416.75 -143.47 -114.60    5.95 1442.68
##
## Coefficients: (2 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2396.092     62.062  38.608  < 2e-16 ***
## xecu            54.749      4.003  13.677  < 2e-16 ***
## xvcpu         -506.793     19.863 -25.515  < 2e-16 ***
## xram            15.749      2.449   6.430 1.45e-10 ***
## xapptype      -839.645     57.870 -14.509  < 2e-16 ***
## xexternalfile   23.813     57.569   0.414    0.679
## xthreading          NA         NA      NA       NA
## xloadinmem          NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 317.7 on 3494 degrees of freedom
## Multiple R-squared:  0.6164, Adjusted R-squared:  0.6158
## F-statistic:  1123 on 5 and 3494 DF,  p-value: < 2.2e-16
```

```r
 predict(ridge.mod ,s=0, exact=T,type="coefficients")[1:8,]
```

```
##  (Intercept)          ecu         vcpu          ram      apptype
##  1990.674452    54.747547  -506.788721    15.748636  -416.573178
## externalfile    threading    loadinmem
##    14.959059    -8.827652  -423.046442
```

choose tuning parameter using cross validation

```
cv.out=cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.out)
```



```
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 33.36339
```

```
ridge.pred=predict(ridge.mod ,s=bestlam ,newx=x[test,])
mean(( ridge.pred -y.test)^2)
```

```
## [1] 93846.97
```

Above code has shown further improvemnt over test MSE so refit the ridge regression model on the full dataset using bestlamda and EXAMINE THE COEFFICIENT ESTIMATES.

```
out=glmnet(x,y,alpha =0)
  predict(out ,type="coefficients",s=bestlam )[1:8,]
```

```
##  (Intercept)          ecu         vcpu          ram      apptype
##  1392.873529     2.688446  -218.187820    -1.697293  -276.041108
## externalfile    threading    loadinmem
##  -119.263954   122.482560  -279.142263
```

## Model diagnostics

Residual plots are the most basic diagnostic graphs when these are plotted gainst the fitted values as well as each of the predictors. Residual plots are used to find: Non linearity in the regression function Non constant variance of error term and if the error terms are not independent Outliers in the data set

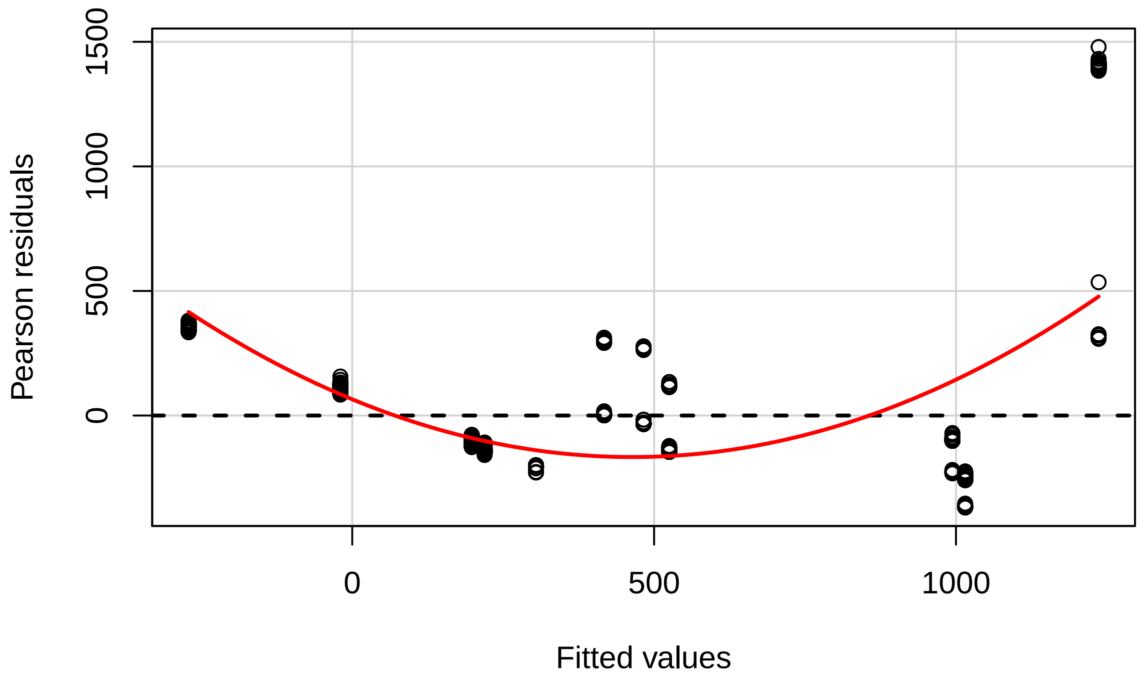
A plot is known as null plot if there is no prominent pattern in it and so it shows the adequacy of model. If a linear model adequately describing the data then the pearson residuals are independ of the fitted values and the predictors and the graphs will be null graphs. No indicative pattern means that the conditional distribution of the residuals (on the vertical axis) should not change with the predictors or fitted values (on x-axis). If the fitted vs pearson residuals has curved general trend it indicates that the model is not adequate to describe the data. Just one null plot is insufficient as an evidence that the model is best fit or adequate however, one non-null plot is sufficient to suggest that the model does not match the data.

This command generates scatter plots of the pearson residulas vs each predictors and the fitted values. By default .the line shown on the graphs are the fitted quadratic regression of the pearson residuals on each predictor. By changing the argument one can plot Studentized residuals instead of pearson. You can even change the quadratic curve to lowess smooth by setting smooth=TRUE and quadratic=FALSE. Write lack of fit test w.r.t each regression model page 289

### Plotting residuals

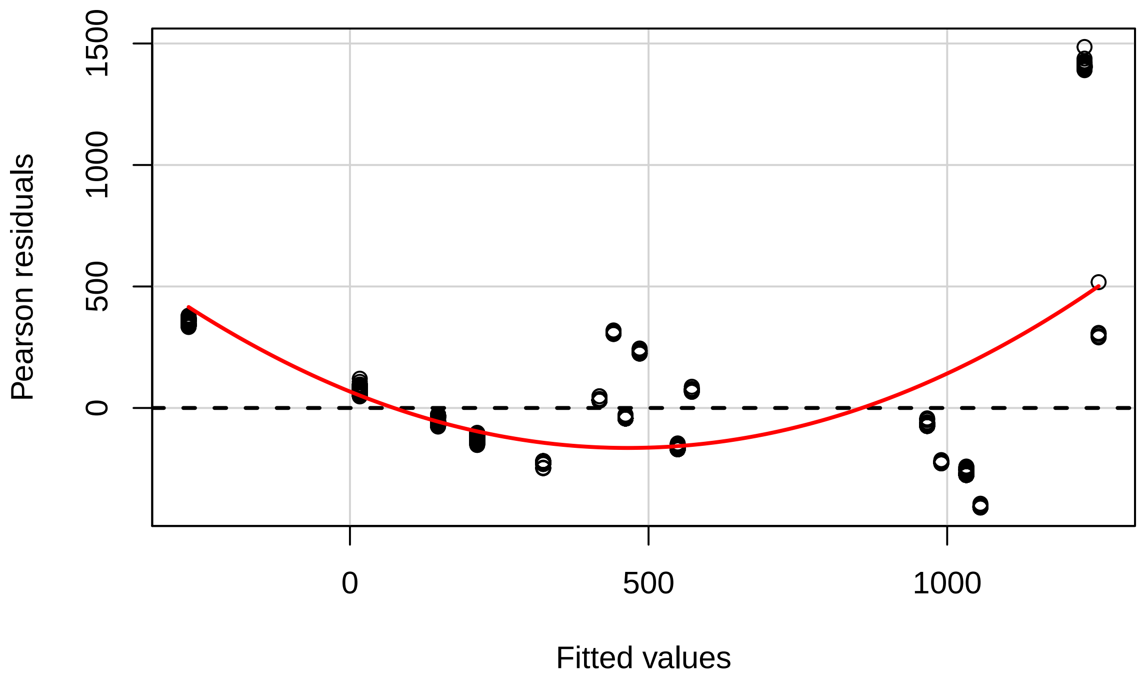
Here we have plotted the graph of residuals vs the fitted values only that can visually describe the adequacy of model fit to data. All of the diagnostic graphs are showing a curve trend as an evidence to indicate that the model is not a best fit for the data or the model does not match the data.

```r
library(car)
residualPlots(fit1, ~1, fitted=TRUE)
```



```
##              Test stat Pr(>|t|)
## Tukey test     81.055        0
```

```r
library(car)
residualPlots(fit2, ~1, fitted=TRUE)
```

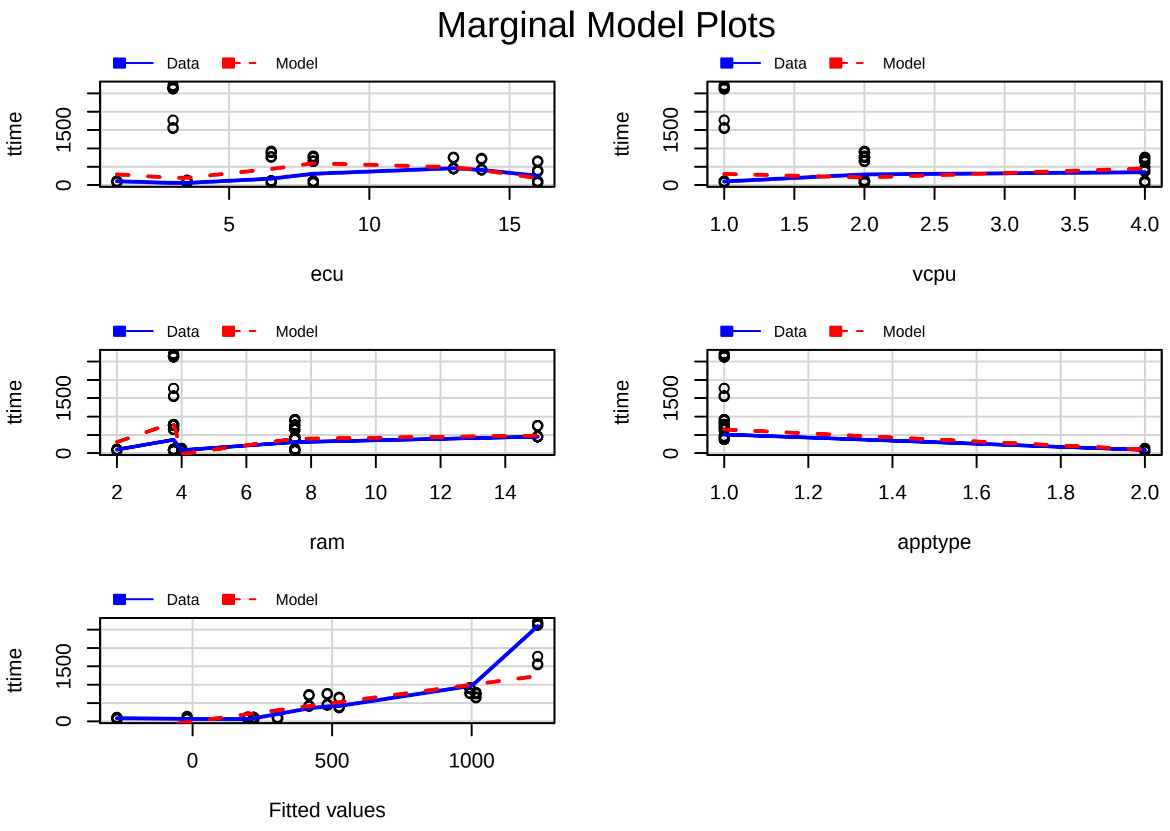


```
##             Test stat Pr(>|t|)
## Tukey test    79.285        0
```

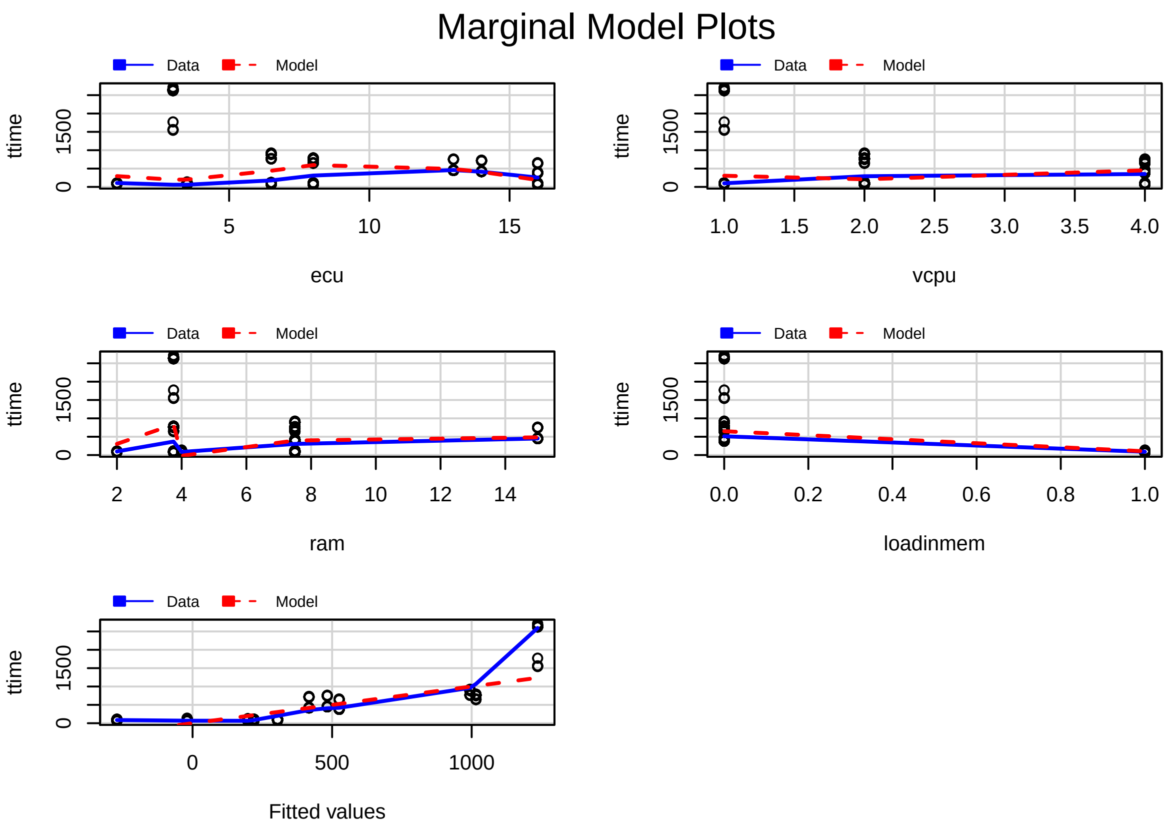
**Marginal model plots**

Marginal model plot is another variation of the basic residual plots in which the response variable is plotted against all the predictors as well as fitted values, in this case ttime is the response variable. The plots of the response vs individual predictor shows the conditional distribution of the response given each predictor and ignoring the other predictors. On the other hand the fiited value vs response variable plot shows the conditional distribution of the response given the fit of the model. In each of the plot, regression function is estimated by fitting a smother to the points in the plot that uses a lowess smooth function and is indicated by a solid line in the graph. The second smooth is also indicated by a dashed line and this line shows that if the fitted values can fairly estimate the response given a predictor on x-axis. If the two lines matches each other it indicates that the model fits the data well, otherwise it indicates the lack of fit. In our case, all the plotted marginal model graphs of, pair of smooths fails to match and provides an evidence that the models are not good fit to data.

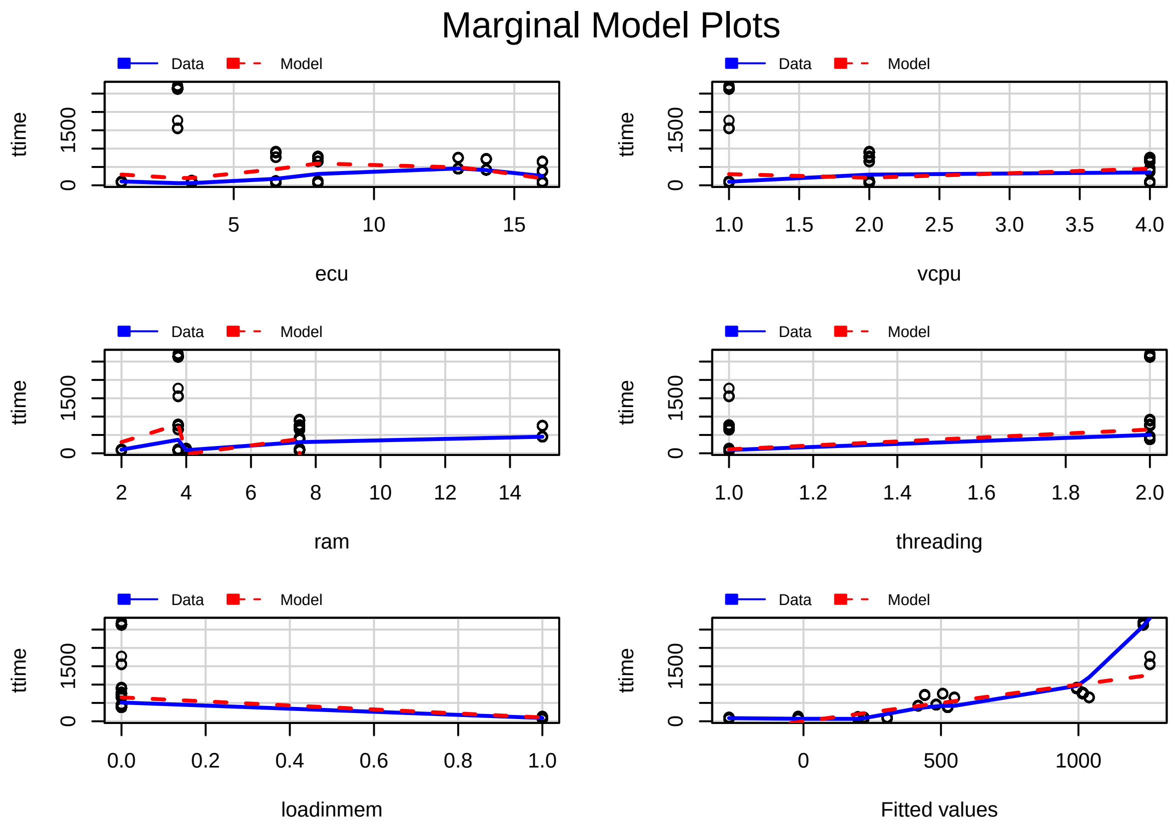
```r
marginalModelPlots(fit1)
```



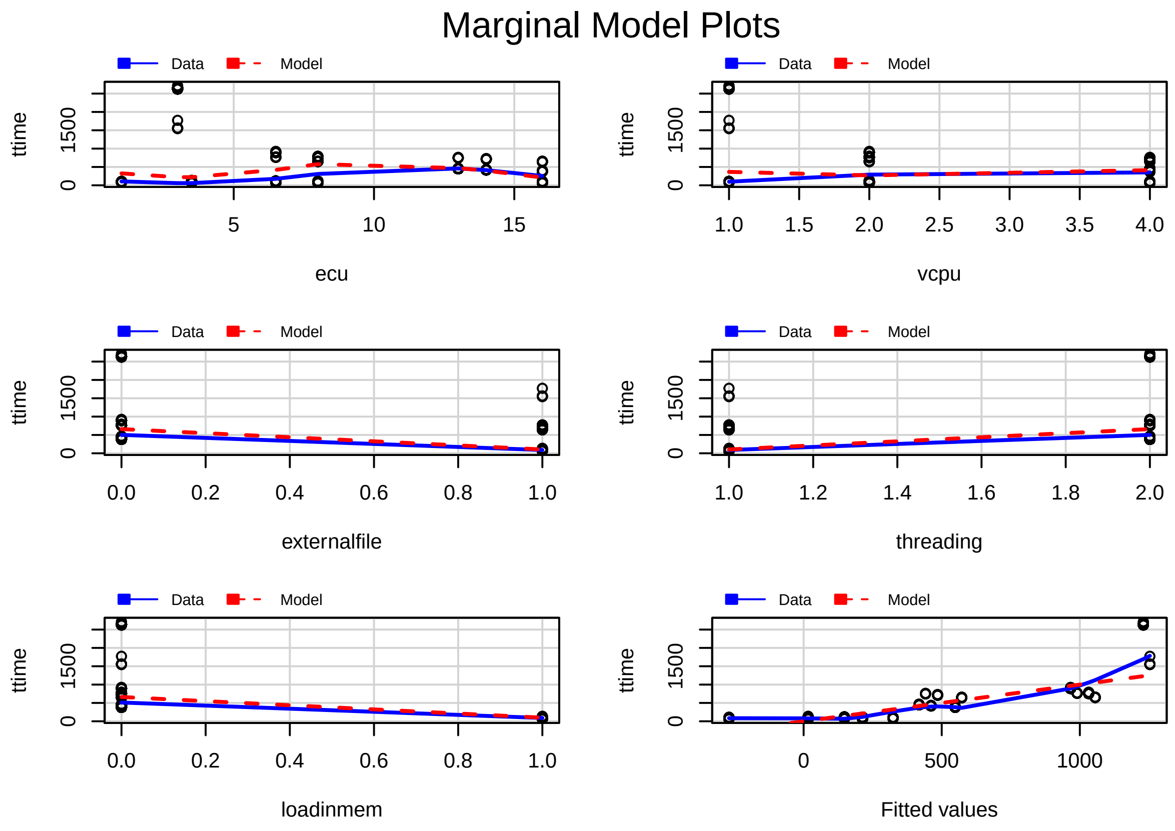
```r
marginalModelPlots(fit2)
```



```
marginalModelPlots(fit3)
```



```r
marginalModelPlots(fit4)
```

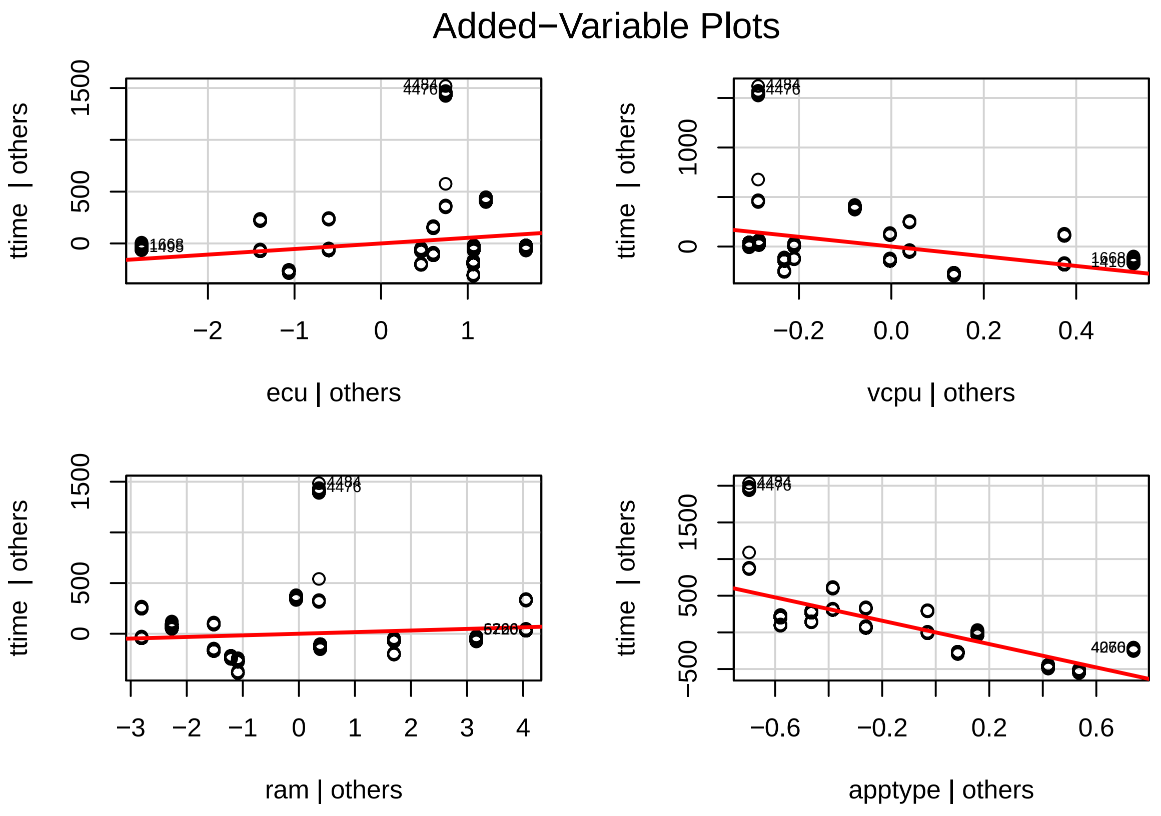


## Added variable plots

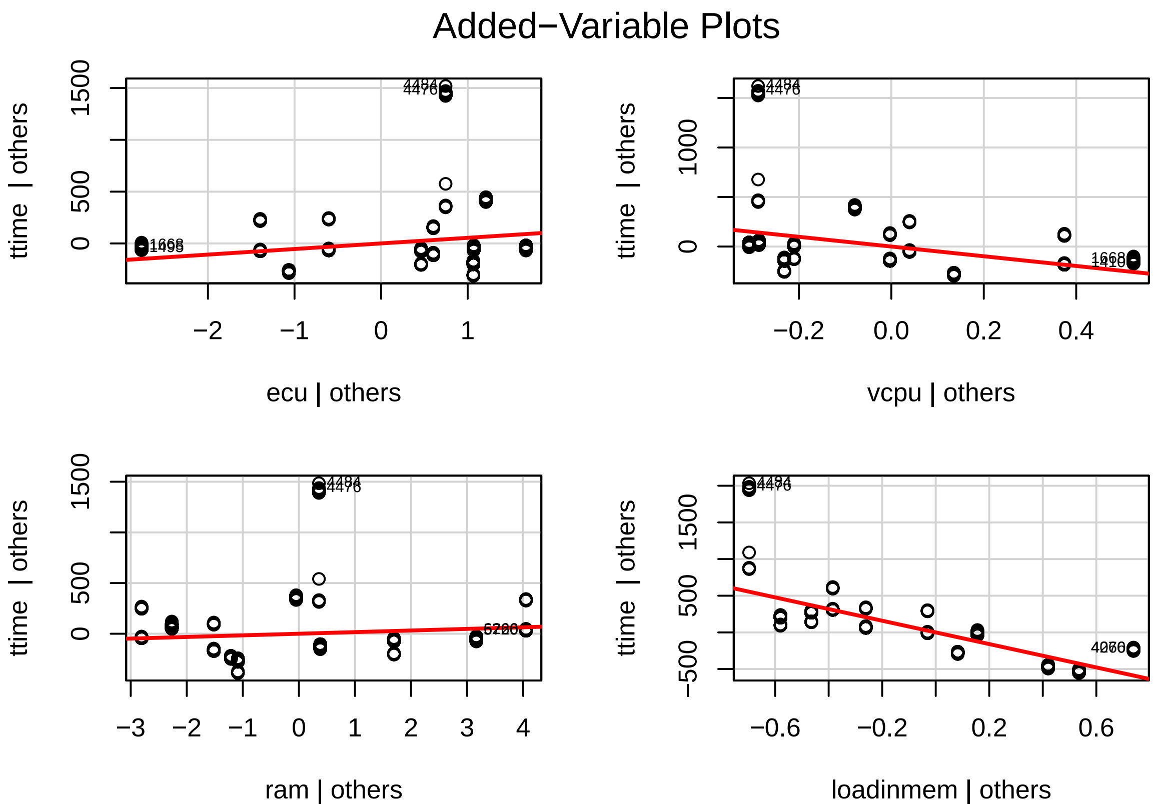
Added variable plot are also called partial regression plots and show the partial relationship between response and predictor adjusted for all other predictors. The graphs are scatterplots in which residuals for y-axis are computed by regressing y on all the regressors excluding x1 while x-axis residulals are obtained by regressing x1 on the other regressors. Added variable graphs are good to see the effect of each regressor after adjusting for all other regressors and shows the impact of observations on regression coefficient.

comment: Leverage plots only work for linear functions.

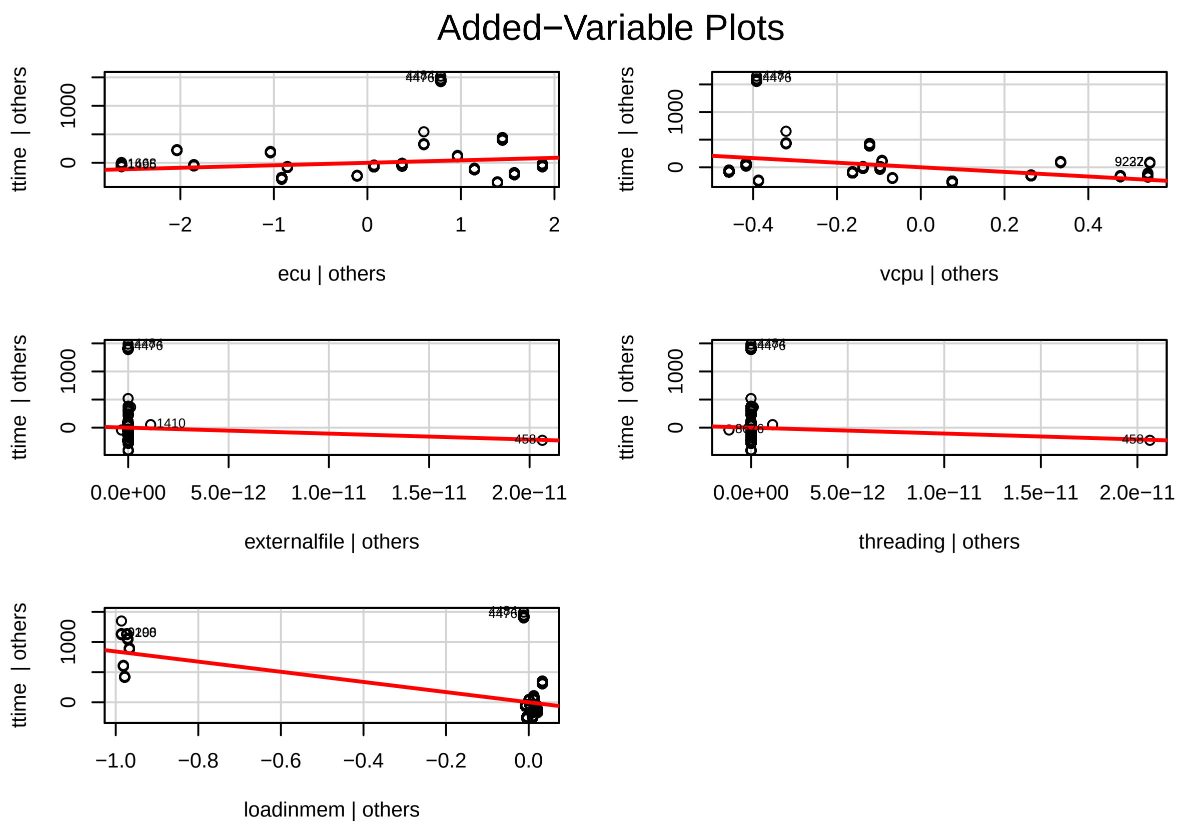
```
avPlots(fit1, id.n=2, id.cex=0.6)
```



```r
avPlots(fit2, id.n=2, id.cex=0.6)
```



```r
avPlots(fit3, id.n=2, id.cex=0.6)
```

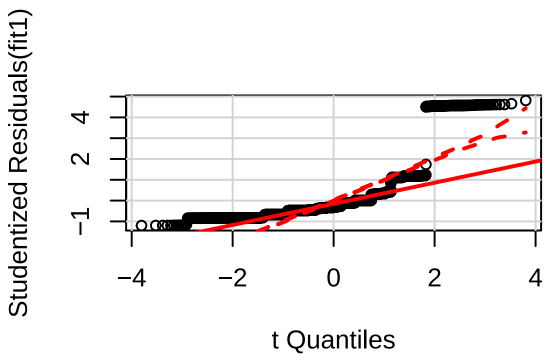


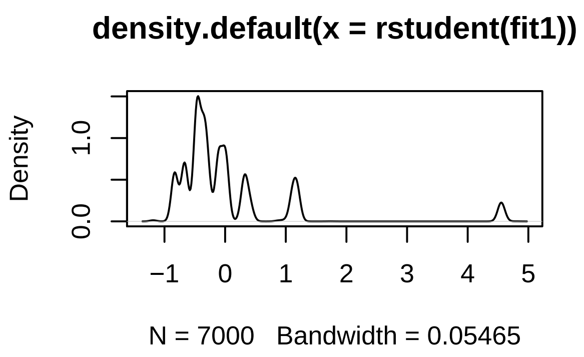
## Transformations

### Response transformation

the generic qqPlot function plots the Studentized residuals against the corresponding quantiles of t(n-k-2). By default this plot generates the 95% pointwise confidence envelope for the Studentized residuals. This graph is useful to check the behaviour of residuals in case of skewness. A non parametric density estimate can figure out the shape of the residual distribution. QQ plot are further used to evaluate. . . . non normal error

```r
par(mfrow=c(2,2))
qqPlot(fit1, id.n=0)
plot(density(rstudent(fit1)))
```





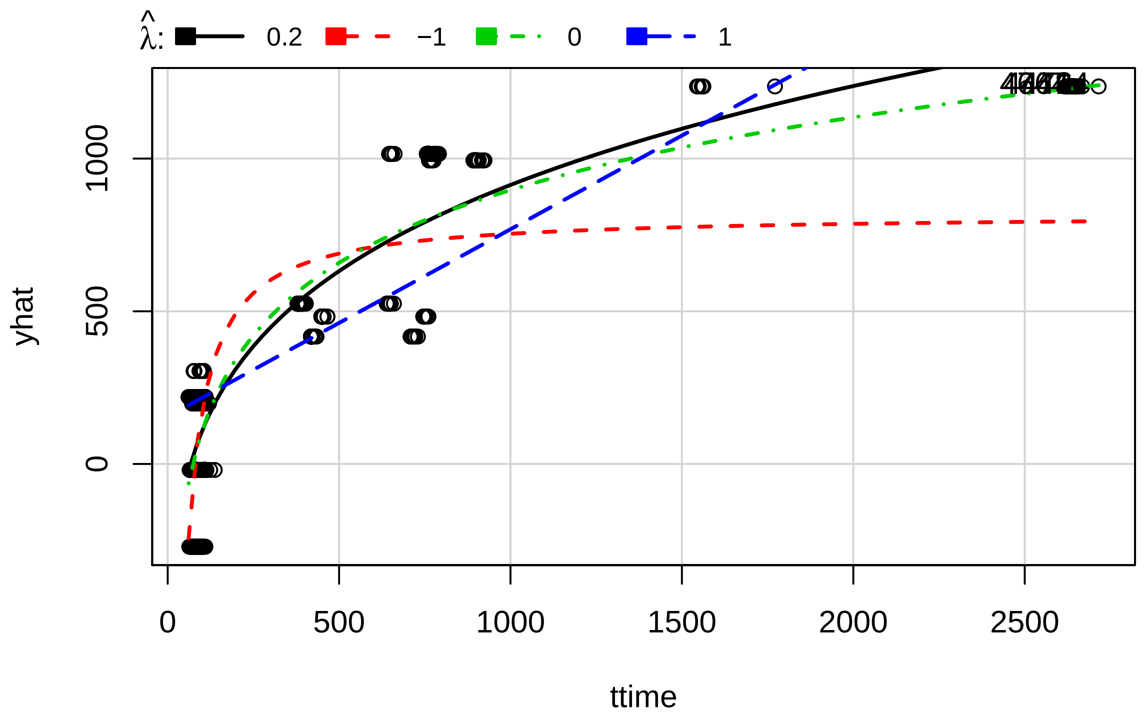
#### Box-Cox transformation Normality transformation: A positive skew in the distribution sometimes can be corrected using the power trnaformation for y. Y (ttime) is strictly positive so we will use Box-Cox power transformation, if y is not poitive one can use Yeo-Johnson function for transformation. poserTransform function in car package provides similar but numeric results compared to boxcox function.
comment: apply qq plot and see. . . ..

```
summary(powerTransform(fit1))
```

```
## bcPower Transformation to Normality
##
##    Est.Power Std.Err. Wald Lower Bound Wald Upper Bound
## Y1   -0.4335   0.0065          -0.4461          -0.4208
##
## Likelihood ratio tests about transformation parameters
##                              LRT df pval
## LR test, lambda = (0)  4297.506  1    0
## LR test, lambda = (1) 27577.265  1    0
```

Linearity transformation: Inverse response plot is an alternative to Box-Cox transformation and this method produces a transformation towards linearity rather than normality.

```
inverseResponsePlot(fit1,id.n=4)
```



```
##       lambda       RSS
## 1  0.1984048 199625859
## 2 -1.0000000 345461903
## 3  0.0000000 209427530
## 4  1.0000000 407141642
```

In our case inverse reponse plot is not successful for selecting a suitable transformation of the response variable. The problem in our case is more likely lack of normality rather than linearity therefore inverse reponse plot is not suitable.

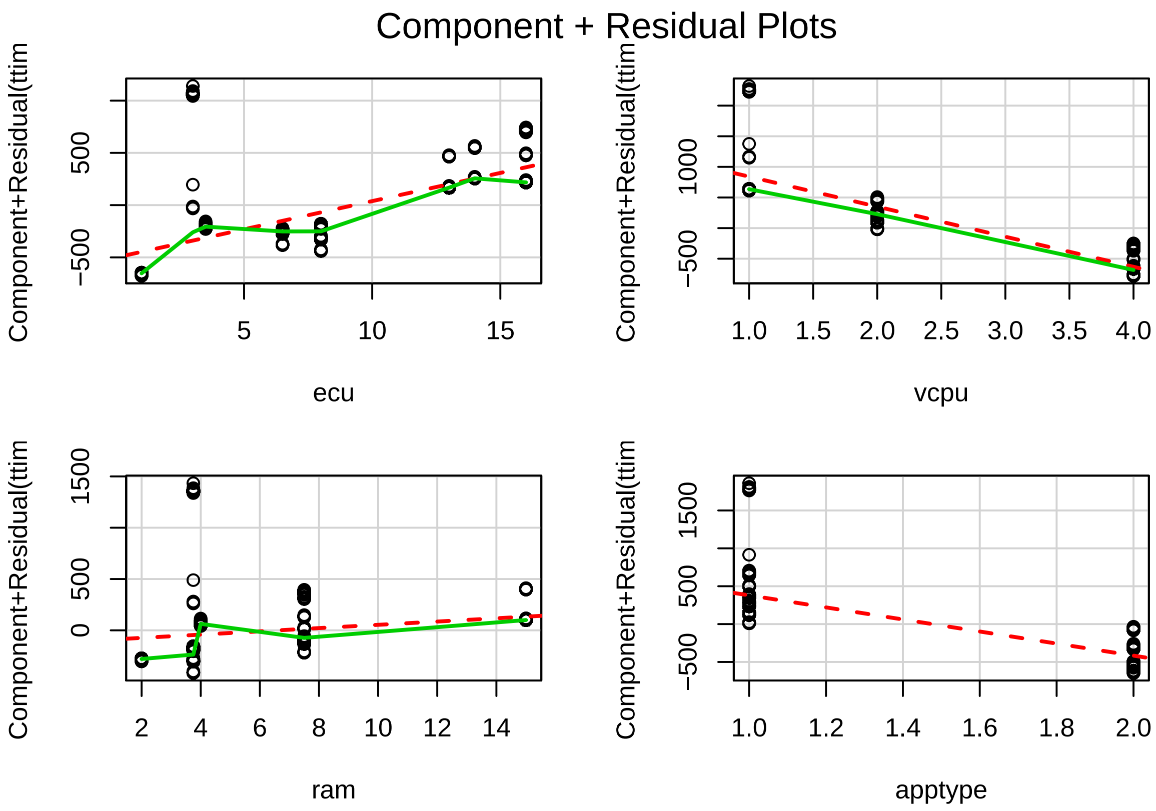
**Predictor transformation**

Sometimes predictor transformation resolves the fitted model problems or reduces the lack of fit. In order to see the non-linearity we evaluate the component-plus-residual plots and help in detectin if the predictor needs some trnasformation.

The outcome of below plots shows that the predictors are not linearly realted to each other.

```
crPlots(fit1)
```

```
## Warning in smoother(.x, partial.res[, var], col = col.lines[2], log.x =
## FALSE, : could not fit smooth
```



we can alter the graphs by permitting quadretic relationship among the predictors and can use following line of code, crPlots(fit1, order=2)

The component-plus-residual plot for ecu looks more like cubic. in contrast the component-plus residual plot for vcpu is slightly non-linear. finally the component-plus residual plot for ram looks like cubic as well.

**The Box-Tidwell method for choosing predictor transformations**: Polyno-mial transformation Linear regression and regularization methods have shown significant limitations in terms of predictive power. This uncertainty of a linear model fit indicated non-linearity in our data that we have diagnosed using crPlot() function. We can relax the linearity assumptions with simple extensions of linear models like polynomial regression.

```
fitp1=lm(ttime~ecu+vcpu+ram+apptype,data=data_train)
summary(fitp1)
```

```
##
## Call:
## lm(formula = ttime ~ ecu + vcpu + ram + apptype, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -369.43 -144.57  -98.06    2.62 1479.33
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2301.675     19.815 116.157   <2e-16 ***
## poly(ecu, 3)3 -52916.950    280.369  -188.7   <2e-16 ***
## vcpu           -1999.328      7.362  -271.6   <2e-16 ***
## poly(ram, 3)1  68302.840    280.487   243.5   <2e-16 ***
## poly(ram, 3)2 -61560.752    287.863  -213.9   <2e-16 ***
## poly(ram, 3)3  67023.229    262.697   255.1   <2e-16 ***
## apptype         -573.041      2.840  -201.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 87.97 on 6991 degrees of freedom
## Multiple R-squared:  0.9685, Adjusted R-squared:  0.9685
## F-statistic: 2.686e+04 on 8 and 6991 DF,  p-value: < 2.2e-16
```

```
fitp11=lm(ttime~poly(ecu,3)+poly(vcpu,2)+poly(ram,3)+apptype,data=data_train)
summary(fitp11)
```

```
##
## Call:
## lm(formula = ttime ~ poly(ecu, 3) + poly(vcpu, 2) + poly(ram,
##     3) + apptype, data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1066.84   -51.92    -4.01    43.79   307.31
##
## Coefficients: (1 not defined because of singularities)
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1247.609      4.325  288.49   <2e-16 ***
## poly(ecu, 3)1  -60359.686    499.403 -120.86   <2e-16 ***
## poly(ecu, 3)2  -73467.397    337.493 -217.69   <2e-16 ***
## poly(ecu, 3)3   51788.600    213.016  243.12   <2e-16 ***
## poly(vcpu, 2)1  67204.918    642.464  104.61   <2e-16 ***
## poly(vcpu, 2)2  81561.468    319.679  255.13   <2e-16 ***
## poly(ram, 3)1  -18923.696    209.502  -90.33   <2e-16 ***
## poly(ram, 3)2    8616.103    152.714   56.42   <2e-16 ***
## poly(ram, 3)3          NA         NA      NA       NA
## apptype          -573.041      2.840 -201.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 87.97 on 6991 degrees of freedom
## Multiple R-squared:  0.9685, Adjusted R-squared:  0.9685
## F-statistic: 2.686e+04 on 8 and 6991 DF,  p-value: < 2.2e-16
```

### Polynomial transformation with Interaction Terms

Alternatively, other functions of the predictors could be considered rather than polynomials. It is not hard to see that there are many possible ways to enlarge the feature space, and that unless we are careful, we could end up with a huge number of features. One might additionally want to enlarge the feature space with higher-order polynomial terms, or with interaction terms.

```r
fitp12=lm(ttime~poly(ecu,3):threading+vcpu+poly(ram,3)+apptype,data=data_train)
summary(fitp12)
```

```
##
## Call:
## lm(formula = ttime ~ poly(ecu, 3):threading + vcpu + poly(ram,
##     3) + apptype, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -449.89  -80.30   -1.42   43.91  725.29
##
## Coefficients:
##                          Estimate Std. Error  t value Pr(>|t|)
## (Intercept)              1805.620     31.163   57.941  < 2e-16 ***
## vcpu                       39.607     11.757    3.369 0.000759 ***
## poly(ram, 3)1            9369.220    453.046   20.681  < 2e-16 ***
## poly(ram, 3)2          -12431.132    435.106  -28.570  < 2e-16 ***
## poly(ram, 3)3           14905.603    347.467   42.898  < 2e-16 ***
## apptype                  -958.800      6.147 -155.989  < 2e-16 ***
## poly(ecu, 3)1:threading -30386.421    576.748  -52.686  < 2e-16 ***
## poly(ecu, 3)2:threading  15667.163    288.543   54.297  < 2e-16 ***
## poly(ecu, 3)3:threading  -2658.827    257.367  -10.331  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 212 on 6991 degrees of freedom
## Multiple R-squared:  0.8171, Adjusted R-squared:  0.8169
## F-statistic:  3904 on 8 and 6991 DF,  p-value: < 2.2e-16
```

```r
fitp13=lm(ttime~poly(ecu,3):threading+poly(vcpu,2)+poly(ram,3)+apptype,data=data_train)
summary(fitp13)
```

```
##
## Call:
## lm(formula = ttime ~ poly(ecu, 3):threading + poly(vcpu, 2) +
##     poly(ram, 3) + apptype, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -974.68  -12.72    0.36    7.81  173.49
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1150.379      2.149  535.25   <2e-16 ***
## poly(vcpu, 2)1         -15762.816    182.391  -86.42   <2e-16 ***
## poly(vcpu, 2)2          53789.569    103.256  520.93   <2e-16 ***
## poly(ram, 3)1           13834.603     72.307  191.33   <2e-16 ***
## poly(ram, 3)2          -20756.764     70.782 -293.25   <2e-16 ***
## poly(ram, 3)3           28258.307     60.739  465.24   <2e-16 ***
## apptype                  -496.702      1.317 -377.01   <2e-16 ***
## poly(ecu, 3)1:threading  -7838.340    101.132  -77.51   <2e-16 ***
## poly(ecu, 3)2:threading -14782.225     74.213 -199.19   <2e-16 ***
## poly(ecu, 3)3:threading   8269.647     45.865  180.30   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33.59 on 6990 degrees of freedom
## Multiple R-squared:  0.9954, Adjusted R-squared:  0.9954
## F-statistic: 1.683e+05 on 9 and 6990 DF,  p-value: < 2.2e-16
```

```
fitp14=lm(ttime~poly(ecu,3):threading+poly(vcpu,2)+poly(ram,3):loadinmem+apptype,data=data_train)
summary(fitp14)
```

```
##
## Call:
## lm(formula = ttime ~ poly(ecu, 3):threading + poly(vcpu, 2) +
##     poly(ram, 3):loadinmem + apptype, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -797.59  -15.45   -2.51   21.09   97.92
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -17331.61     325.98  -53.17   <2e-16 ***
## poly(vcpu, 2)1            -7825.45     210.06  -37.25   <2e-16 ***
## poly(vcpu, 2)2            41755.99     249.49  167.37   <2e-16 ***
## apptype                   18211.86     327.75   55.57   <2e-16 ***
## poly(ecu, 3)1:threading  -17628.09     203.18  -86.76   <2e-16 ***
## poly(ecu, 3)2:threading   -5186.41     197.58  -26.25   <2e-16 ***
## poly(ecu, 3)3:threading    5651.37      78.73   71.78   <2e-16 ***
## poly(ram, 3)1:loadinmem 3785642.68   65637.69   57.67   <2e-16 ***
## poly(ram, 3)2:loadinmem 2550051.95   44714.94   57.03   <2e-16 ***
## poly(ram, 3)3:loadinmem  408182.07    6605.98   61.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29.62 on 6990 degrees of freedom
## Multiple R-squared:  0.9964, Adjusted R-squared:  0.9964
## F-statistic: 2.167e+05 on 9 and 6990 DF,  p-value: < 2.2e-16
```

### Nonconstant error variance

Cross Validation–Model Selection Model assessment was done using cross-validation using the MSE which estimated the test errors associated with the learning method to evaluate its performance. Most commonly used measure for the quality of fit in regression analysis is the mean squared error (MSE) and cross validation uses this measure. In our case we preferred to choose the model that is lower in test MSE compared to training MSE. We applied regression diagnostics to check the assumptions for linear regression with non-linear transformation of the predictors. Non-linear models outperformed the linear ones as indicated by different factors. Our prediction function is showing the similar results that we derived from data visualisation this shows the validation of our prediction method.

```r
library(boot)
glm.fit14=glm(ttime~poly(ecu,3):threading+poly(vcpu,2)+poly(ram,3):loadinmem+apptype,data=data_train)
cv.error=cv.glm(data_train,glm.fit14,K=10)$delta[1]
cv.error
```

```
## [1] 897.0517
```

```r
mean((newdata$ttime-predict(glm.fit14,newdata))[-train1]^2)
```

```
## [1] 661.2765
```

# Bibliography

[1] Statista, Cloud computing market view, `https://www.statista.com/statistics/475670/cloud-applications-market-size-worldwide/` (Accessed: 2018-03-22).

[2] Gartner, Cloud computing market view, `https://www.gartner.com/newsroom/id/3815165` (Accessed: 2018-03-22).

[3] A. Barker, B. Varghese, J. S. Ward, I. Sommerville, Academic cloud computing research: Five pitfalls and five opportunities, in: Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing, HotCloud'14, USENIX Association, Berkeley, CA, USA, 2014, pp. 2–2.
URL `http://dl.acm.org/citation.cfm?id=2696535.2696537`

[4] P. Leitner, J. Cito, Patterns in the chaos - a study of performance variation and predictability in public iaas clouds, CoRR abs/1411.2429.
URL `http://arxiv.org/abs/1411.2429`

[5] F. Samreen, G. S. Blair, M. Rowe, Adaptive decision making in multi-cloud management, in: 2nd International Workshop on CrossCloud Systems, ACM, 2014, pp. 4:1–4:6. doi:10.1145/2676662.2676676.

[6] P. Leitner, J. Cito, Patterns in the chaos — a study of performance variation and predictability in public IaaS clouds, ACM Transactions on Internet Technology 16 (3) (2016) 15:1–15:23. doi:10.1145/2885497.

[7] Z. Ou, H. Zhuang, J. K. Nurminen, A. Ylä-Jääski, P. Hui, Exploiting hardware heterogeneity within the same instance type of amazon ec2, in: HotCloud, USENIX, 2012.
URL `https://www.usenix.org/conference/hotcloud12/exploiting-hardware-heterogeneity-wit`

[8] Y. Elkhatib, Mapping Cross-Cloud Systems: Challenges and Opportunities, in: Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing, USENIX Association, 2016, pp. 77–83.

[9] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, in: 2008 Grid Computing Environments Workshop, 2008, pp. 1–10. doi:10.1109/GCE.2008.4738445.

[10] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, Journal of Internet Services and Applications 1 (1) (2010) 7–18. doi:10.1007/s13174-010-0007-6. URL http://dx.doi.org/10.1007/s13174-010-0007-6

[11] I. S. Artan Mazrekaj, B. Sejdiu, Pricing schemes in cloud computing: An overview, International Journal of Advanced Computer Science and Applications(ijacsa),7(2), 2016. URL http://dx.doi.org/10.14569/IJACSA.2016.070211

[12] S. Grivas, T. Kumar, H. Wache, Cloud broker: Bringing intelligence into the cloud, in: IEEE International Conference on Cloud Computing (CLOUD), 2010, pp. 544–545. doi:10.1109/CLOUD.2010.48.

[13] P. M. Mell, T. Grance, Sp 800-145. the nist definition of cloud computing, Tech. rep., Gaithersburg, MD, United States (2011).

[14] A. Barker, B. Varghese, L. Thai, Cloud services brokerage: A survey and research roadmap, CoRR abs/1506.00485. URL http://arxiv.org/abs/1506.00485

[15] RightScale), Rightscale cloud portfolio management, http://rightscale.com/why-cloud-management-platform/benefits.html (2016).

[16] enStratus, enstratus, http://enstratus.html (2016).

[17] xStream, xstream from virtustream, http://virtustream.com/software.html (2016).

[18] CliQr, Cliqr, http://cliqr.html (2016).

[19] Apache Brooklyn, Apache brooklyn, https://brooklyn.apache.org/.html (2016).

[20] Scalr, Scalr the cloud management platform, http://.scalr.com.html (2016).

[21] Standingcloud by AppDirect, Cloud application marketplace and management platform, http://www.standingcloud.com.html (2016).

[22] The Aeloud Project, Aelous, http://www.aeolus-project.org.html (2016).

[23] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, S. Mankovski, Introducing stratos: A cloud broker service, in: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, 2012, pp. 891–898. doi:10.1109/CLOUD.2012.24.

[24] S. Gupta, V. Muntes-Mulero, P. Matthews, J. Dominiak, A. Omerovic, J. Aranda, S. Seycek, Risk-driven framework for decision support in cloud service selection, in: International Symposium on Cluster, Cloud and Grid Computing (CCGrid), IEEE/ACM, 2015, pp. 545–554. doi:10.1109/CCGrid.2015.111.

[25] E. Kamateri, N. Loutas, D. Zeginis, J. Ahtes, F. D'Andria, S. Bocconi, P. Gouvas, G. Ledakis, F. Ravagli, O. Lobunets, K. Tarabanis, Cloud4SOA: A Semantic-Interoperability PaaS Solution for Multi-cloud Platform Management and Portability, Vol. 8135 of LNCS, Springer, 2013, Ch. 6, pp. 64–78. doi:10.1007/978-3-642-40651-5_6.

[26] A. Li, X. Yang, S. Kandula, M. Zhang, Cloudcmp: Comparing public cloud providers, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC '10, ACM, New York, NY, USA, 2010, pp. 1–14. doi:10.1145/1879141.1879143.
URL http://doi.acm.org/10.1145/1879141.1879143

[27] I. Patiniotakis, Y. Verginadis, G. Mentzas, Preference-based cloud service recommendation as a brokerage service, in: 2nd International Workshop on CrossCloud Systems, ACM, 2014, pp. 5:1–5:6. doi:10.1145/2676662.2676677.

[28] A. Li, X. Zong, S. Kandula, X. Yang, M. Zhang, Cloudprophet: Towards application performance prediction in cloud, in: Proceedings of the ACM SIGCOMM 2011 Conference, SIGCOMM '11, ACM, New York, NY, USA, 2011, pp. 426–427. doi:10.1145/2018436.2018502.
URL http://doi.acm.org/10.1145/2018436.2018502

[29] B. Varghese, O. Akgun, I. Miguel, L. Thai, A. Barker, Cloud benchmarking for performance, in: 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, 2014, pp. 535–540. doi:10.1109/CloudCom.2014.28.

[30] R. C. Chiang, J. Hwang, H. H. Huang, T. Wood, Matrix: Achieving predictable virtual machine performance in the clouds, in: 11th International Conference on Autonomic Computing (ICAC 14), USENIX Association, Philadelphia, PA, 2014, pp. 45–56.
URL https://www.usenix.org/conference/icac14/technical-sessions/presentation/chiang

[31] I. Patiniotakis, Y. Verginadis, G. Mentzas, Preference-based cloud service recommendation as a brokerage service, in: Proceedings of the 2Nd International Workshop on CrossCloud Systems,

CCB '14, ACM, New York, NY, USA, 2014, pp. 5:1–5:6. doi:10.1145/2676662.2676677.
URL http://doi.acm.org/10.1145/2676662.2676677

[32] E. Feller, L. Ramakrishnan, C. Morin, On the Performance and Energy Efficiency of Hadoop Deployment Models, in: The IEEE International Conference on Big Data 2013 (IEEE BigData 2013), Santa Clara, United States, 2013.
URL https://hal.inria.fr/hal-00856330

[33] D. Petcu, Multi-cloud: Expectations and current approaches, in: Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds, MultiCloud '13, ACM, New York, NY, USA, 2013, pp. 1–6. doi:10.1145/2462326.2462328.
URL http://doi.acm.org/10.1145/2462326.2462328

[34] D. Guyon, A. C. Orgerie, C. Morin, Energy-efficient user-oriented cloud elasticity for data-driven applications, in: 2015 IEEE International Conference on Data Science and Data Intensive Systems, 2015, pp. 376–383. doi:10.1109/DSDIS.2015.57.

[35] D. Margery, D. Guyon, A.-C. Orgerie, C. Morin, G. Francis, C. Palansuriya, K. Kavoussanakis, A CO2 emissions accounting framework with market-based incentives for Cloud infrastructures, in: SMARTGREENS: International Conference on Smart Cities and Green ICT Systems, Porto, Portugal, 2017.
URL https://hal.inria.fr/hal-01486185

[36] A. J. Ferrer, F. HernáNdez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgó, T. Sharif, C. Sheridan, Optimis: A holistic approach to cloud service provisioning, Future Gener. Comput. Syst. 28 (1) (2012) 66–77. doi:10.1016/j.future.2011.05.022.
URL http://dx.doi.org/10.1016/j.future.2011.05.022

[37] D. Petcu, Portability and interoperability between clouds: Challenges and case study, in: Proceedings of the 4th European Conference on Towards a Service-based Internet, ServiceWave'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 62–74.
URL http://dl.acm.org/citation.cfm?id=2050869.2050876

[38] G. Blair, D. Schmidt, C. Taconet, Middleware for internet distribution in the context of cloud computing and the internet of things, Annals of Telecommunications 71 (3) (2016) 87–92.

doi:10.1007/s12243-016-0493-z.
URL http://dx.doi.org/10.1007/s12243-016-0493-z

[39] F. Samreen, Y. Elkhatib, M. Rowe, G. S. Blair, Daleel: Simplifying cloud instance selection using machine learning, in: Proceedings of the Network Operations and Management Symposium, IEEE, 2016, pp. 557–563. doi:10.1109/NOMS.2016.7502858.

[40] A. K. Maji, S. Mitra, B. Zhou, S. Bagchi, A. Verma, Mitigating interference in cloud services by middleware reconfiguration, in: Proceedings of the 15th International Middleware Conference, Middleware '14, ACM, 2014, pp. 277–288. doi:10.1145/2663165.2663330.

[41] N. Grozev, R. Buyya, Inter-cloud architectures and application brokering: Taxonomy and survey, Softw. Pract. Exper. 44 (3) (2014) 369–390. doi:10.1002/spe.2168.
URL http://dx.doi.org/10.1002/spe.2168

[42] L. Badger, R. Bohn, S. Chu, M. Hogan, F. Liu, V. Kaufmann, J. Mao, J. Messina, K. Mills, A. Sokol, J. Tong, F. Whiteside, D. Leaf, US government cloud computing technology roadmap, Tech. Rep. (Special Publication) 500-293, National Institute of Standards and Technology (Nov. 2011).

[43] chef: Automate your infrastructure, https://www.chef.io/chef/.

[44] Configuration management with puppet, https://puppet.com/solutions/configuration-management.

[45] Ansible for configuration management, https://www.ansible.com/.

[46] Cfengine automation tool, https://cfengine.com/.

[47] Saltstack automation for cloudops, https://saltstack.com/.

[48] Apache jcloud, https://jclouds.apache.org/.

[49] Apache libcloud api, https://libcloud.apache.org/.

[50] Apache deltacloud api, https://deltacloud.apache.org/.

[51] Simple cloud api, https://simplecloud.org/.

[52] Kavoo), Kavoo, http://kavoo.com (2016).

[53] CELAR, Celar, http://linc.ucy.ac.cy/CELAR/ (2016).

[54] D. Ardagna, E. D. Nitto, P. Mohagheghi, S. Mosser, C. Ballagny, F. D'Andria, G. Casale, P. Matthews, C. S. Nechifor, D. Petcu, A. Gericke, C. Sheridan, Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds, in: 2012 4th International Workshop on Modeling in Software Engineering (MISE), 2012, pp. 50–56. doi:10.1109/MISE.2012.6226014.

[55] A. E. Gencer, D. Bindel, E. G. Sirer, R. van Renesse, Configuring distributed computations using response surfaces, in: Proceedings of the 16th Annual Middleware Conference, Middleware '15, ACM, New York, NY, USA, 2015, pp. 235–246. doi:10.1145/2814576.2814730.
URL http://doi.acm.org/10.1145/2814576.2814730

[56] S. Sundareswaran, A. Squicciarini, D. Lin, A brokerage-based approach for cloud service selection, in: IEEE International Conference on Cloud Computing (CLOUD), 2012, pp. 558–565. doi:10.1109/CLOUD.2012.119.

[57] J. Siegel, J. Perdue, Cloud services measures for global use: The service measurement index (smi), in: 2012 Annual SRII Global Conference, 2012, pp. 411–415. doi:10.1109/SRII.2012.51.

[58] S. K. Garg, S. Versteeg, R. Buyya, A framework for ranking of cloud computing services, Future Gener. Comput. Syst. 29 (4) (2013) 1012–1023. doi:10.1016/j.future.2012.06.006.
URL http://dx.doi.org/10.1016/j.future.2012.06.006

[59] G. Baranwal, D. P. Vidyarthi, A cloud service selection model using improved ranked voting method, Concurrency and Computation: Practice and Experience 28 (13) (2016) 3540–3567, cPE-14-0389.R2. doi:10.1002/cpe.3740.
URL http://dx.doi.org/10.1002/cpe.3740

[60] Key Performance Indicators for Cloud Computing SLAs, ACM, New York, NY, USA, 2009, 534095.

[61] O. Aida, Supporting cloud service selection with a risk-driven cost-benefit analysis, in: Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2015, Taormina, Italy, September 15-17, 2015, Revised Selected Papers, Springer International Publishing, Cham, 2016, pp. 166–174.
URL http://dx.doi.org/10.1007/978-3-319-33313-7_12

[62] B. Balis, K. Figiela, A Lightweight Approach for Deployment of Scientific Workflows in Cloud Infrastructures, 2016.

[63] A. Menychtas, K. Konstanteli, J. Alonso, L. Orue-Echevarria, J. Gorronogoitia, G. Kousiouris, C. Santzaridou, H. Bruneliere, B. Pellens, P. Stuer, et al., Software modernization and cloudification using the ARTIST migration methodology and framework, Scalable Computing: Practice and Experience 15 (2) (2014) 131–152. doi:10.12694/scpe.v15i2.980.

[64] N. Kaviani, E. Wohlstadter, R. Lea, Profiling-as-a-service: Adaptive scalable resource profiling for the cloud in the cloud, in: G. Kappel, Z. Maamar, H. R. Motahari-Nezhad (Eds.), Service-Oriented Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 157–171.

[65] A. Wieder, P. Bhatotia, A. Post, R. Rodrigues, Orchestrating the deployment of computations in the cloud with conductor, in: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, NSDI'12, USENIX Association, Berkeley, CA, USA, 2012, pp. 27–27.
URL `http://dl.acm.org/citation.cfm?id=2228298.2228335`

[66] T. Mitchell, Machine Learning, McGraw Hill, 1997.

[67] N. N. Pise, P. Kulkarni, A survey of semi-supervised learning methods, in: 2008 International Conference on Computational Intelligence and Security, Vol. 2, 2008, pp. 30–34. doi:10.1109/CIS.2008.204.

[68] V. J. Prakash, L. M. Nithya, A survey on semi-supervised learning techniques, CoRR abs/1402.4645. arXiv:1402.4645.
URL `http://arxiv.org/abs/1402.4645`

[69] S. B. Kotsiantis, Supervised machine learning: A review of classification techniques, in: Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, IOS Press, Amsterdam, The Netherlands, The Netherlands, 2007, pp. 3–24.
URL `http://dl.acm.org/citation.cfm?id=1566770.1566773`

[70] P. Mehta, H. Shah, V. Kori, V. Vikani, S. Shukla, M. Shenoy, Survey of unsupervised machine learning algorithms on precision agricultural data, in: 2015 International Conference on

Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015, pp. 1–8. doi:10.1109/ICIIECS.2015.7193070.

[71] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: A survey, CoRR cs.AI/9605103.
URL `http://arxiv.org/abs/cs.AI/9605103`

[72] J. Kupferman, J. Silverman, P. Jara, J. Browne, Scaling into the cloud, Tech. rep., University of California, Santa Barbara (2009).

[73] E. Caron, F. Desprez, A. Muresan, Forecasting for grid and cloud computing on-demand resources based on pattern matching, in: Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2010, pp. 456–463. doi:10.1109/CloudCom.2010.65.

[74] A. A. Bankole, S. A. Ajila, Predicting cloud resource provisioning using machine learning techniques, in: Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE, 2013, pp. 1–4. doi:10.1109/CCECE.2013.6567848.

[75] S. Islam, J. Keung, K. Lee, A. Liu, Empirical prediction models for adaptive resource provisioning in the cloud, Future Generation Computer Systems 28 (1) (2012) 155–162. doi:http://dx.doi.org/10.1016/j.future.2011.05.027.

[76] J. Gao, Machine learning application for data center optimization, Tech. rep., Google (2013).

[77] C.-T. G. H.-M. C. :Chenn-Jung. Huang, Yu-Wu. Wang, J.-J. Jian, Applications of machine learning to resource management in cloud computing, 2013.

[78] J. Hadley, Y. Elkhatib, G. S. Blair, U. Roedig, Metabox: A lightweight, vendor-independent cloud broker, in: Under submission to Middleware, 2015.

[79] E. Alpaydin, Introduction to Machine Learning, MIT Press, 2014.

[80] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning: With Applications in R, 4th Edition, Springer, 2014.

[81] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning, seventh Edition, Springer, 2013.

[82] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and Computing 14 (3) (2004) 199–222. doi:10.1023/B:STCO.0000035301.49549.88.
URL http://dx.doi.org/10.1023/B:STCO.0000035301.49549.88

[83] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, V. Vapnik, Predicting time series with support vector machines, in: Proceedings of the 7th International Conference on Artificial Neural Networks, ICANN '97, Springer-Verlag, London, UK, UK, 1997, pp. 999–1004.
URL http://dl.acm.org/citation.cfm?id=646257.685538

[84] A. Baron, P. Rayson, VARD2: A tool for dealing with spelling variation in historical corpora, in: Postgraduate conference in corpus linguistics, 2008.

[85] Itemrecommender: Movielens dataset, https://grouplens.org/datasets/movielens/.

[86] RightScale, 2015 state of the cloud survey, Tech. rep. (2015).

[87] H. Shimodaira, Improving predictive inference under covariate shift by weighting the log-likelihood function (2000).

[88] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. on Knowl. and Data Eng. 22 (10) (2010) 1345–1359. doi:10.1109/TKDE.2009.191.
URL http://dx.doi.org/10.1109/TKDE.2009.191

[89] K. Weiss, T. M. Khoshgoftaar, D. Wang, A survey of transfer learning, Journal of Big Data 3 (1) (2016) 9. doi:10.1186/s40537-016-0043-6.
URL http://dx.doi.org/10.1186/s40537-016-0043-6

[90] Y. Zhu, Y. Chen, Z. Lu, S. J. Pan, G.-R. Xue, Y. Yu, Q. Yang, Heterogeneous transfer learning for image classification, in: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI'11, AAAI Press, 2011, pp. 1304–1309.
URL http://dl.acm.org/citation.cfm?id=2900423.2900630

[91] W. Li, L. Duan, D. Xu, I. W. Tsang, Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (6) (2014) 1134–1148. doi:10.1109/TPAMI.2013.167.

[92] B. Kulis, K. Saenko, T. Darrell, What you saw is not what you get: Domain adaptation using asymmetric kernel transforms, in: Proceedings of the 2011 IEEE Conference on Computer Vision

and Pattern Recognition, CVPR '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 1785–1792. doi:10.1109/CVPR.2011.5995702.
URL `http://dx.doi.org/10.1109/CVPR.2011.5995702`

[93] C. Wang, S. Mahadevan, Heterogeneous domain adaptation using manifold alignment, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two, IJCAI'11, AAAI Press, 2011, pp. 1541–1546. doi:10.5591/978-1-57735-516-8/IJCAI11-259.
URL `http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-259`

[94] M. Gal-on, S. Mannor, Learning from multiple outlooks, CoRR abs/1005.0027.
URL `http://arxiv.org/abs/1005.0027`

[95] J. Nam, S. Kim, Heterogeneous defect prediction, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, ACM, New York, NY, USA, 2015, pp. 508–519. doi:10.1145/2786805.2786814.
URL `http://doi.acm.org/10.1145/2786805.2786814`

[96] Y. R. Yeh, C. H. Huang, Y. C. F. Wang, Heterogeneous domain adaptation and classification by exploiting the correlation subspace, IEEE Transactions on Image Processing 23 (5) (2014) 2009–2018. doi:10.1109/TIP.2014.2310992.

[97] J. T. Zhou, S. J. Pan, I. W. Tsang, Y. Yan, Hybrid heterogeneous transfer learning through deep learning, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI'14, AAAI Press, 2014, pp. 2213–2219.
URL `http://dl.acm.org/citation.cfm?id=2892753.2892859`

[98] P. Prettenhofer, B. Stein, Cross-language text classification using structural correspondence learning, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, Association for Computational Linguistics, Stroudsburg, PA, USA, 2010, pp. 1118–1127.
URL `http://dl.acm.org/citation.cfm?id=1858681.1858795`

[99] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, T. G. Dietterich, To transfer or not to transfer, in: In NIPS'05 Workshop, Inductive Transfer: 10 Years Later, 2005.

[100] R. Chattopadhyay, Q. Sun, W. Fan, I. Davidson, S. Panchanathan, J. Ye, Multisource domain adaptation and its application to early detection of fatigue, ACM Trans. Knowl. Discov. Data 6 (4) (2012) 18:1–18:26. doi:10.1145/2382577.2382582.
URL http://doi.acm.org/10.1145/2382577.2382582

[101] B. Gong, Y. Shi, F. Sha, K. Grauman, Geodesic flow kernel for unsupervised domain adaptation, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 2066–2073. doi:10.1109/CVPR.2012.6247911.

[102] R. Caruana, Multitask learning, Machine Learning 28 (1) (1997) 41–75. doi:10.1023/A:1007379606734.
URL http://dx.doi.org/10.1023/A:1007379606734

[103] E. V. Bonilla, K. M. Chai, C. Williams, Multi-task gaussian process prediction, in: J. C. Platt, D. Koller, Y. Singer, S. T. Roweis (Eds.), Advances in Neural Information Processing Systems 20, Curran Associates, Inc., 2008, pp. 153–160.
URL http://papers.nips.cc/paper/3189-multi-task-gaussian-process-prediction.pdf

[104] T. Evgeniou, M. Pontil, Regularized multi–task learning, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04, ACM, New York, NY, USA, 2004, pp. 109–117. doi:10.1145/1014052.1014067.
URL http://doi.acm.org/10.1145/1014052.1014067

[105] R. Raina, A. Battle, H. Lee, B. Packer, A. Y. Ng, Self-taught learning: Transfer learning from unlabeled data, in: Proceedings of the 24th International Conference on Machine Learning, ICML '07, ACM, New York, NY, USA, 2007, pp. 759–766. doi:10.1145/1273496.1273592.
URL http://doi.acm.org/10.1145/1273496.1273592

[106] H. D. III, D. Marcu, Domain adaptation for statistical classifiers, CoRR abs/1109.6341.
URL http://arxiv.org/abs/1109.6341

[107] A. Arnold, R. Nallapati, W. Cohen, A comparative study of methods for transductive transfer learning, in: Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on, 2007, pp. 77–82. doi:10.1109/ICDMW.2007.109.

[108] H. Daumé, III, A. Kumar, A. Saha, Frustratingly easy semi-supervised domain adaptation, in: Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing, DANLP 2010, Association for Computational Linguistics, Stroudsburg, PA, USA, 2010, pp. 53–59.
URL http://dl.acm.org/citation.cfm?id=1870526.1870534

[109] W. Dai, Q. Yang, G.-R. Xue, Y. Yu, Boosting for transfer learning, in: Proceedings of the 24th International Conference on Machine Learning, ICML '07, ACM, New York, NY, USA, 2007, pp. 193–200. doi:10.1145/1273496.1273521.
URL http://doi.acm.org/10.1145/1273496.1273521

[110] W. Dai, G.-R. Xue, Q. Yang, Y. Yu, Transferring naive bayes classifiers for text classification, in: Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1, AAAI'07, AAAI Press, 2007, pp. 540–545.
URL http://dl.acm.org/citation.cfm?id=1619645.1619732

[111] J. Jiang, C. Zhai, Instance weighting for domain adaptation in nlp, in: In ACL 2007, 2007, pp. 264–271.

[112] B. Zadrozny, Learning and evaluating classifiers under sample selection bias, in: Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, ACM, New York, NY, USA, 2004, pp. 114–. doi:10.1145/1015330.1015425.
URL http://doi.acm.org/10.1145/1015330.1015425

[113] X. Liao, Y. Xue, L. Carin, Logistic regression with an auxiliary data source, in: Proceedings of the 22Nd International Conference on Machine Learning, ICML '05, ACM, New York, NY, USA, 2005, pp. 505–512. doi:10.1145/1102351.1102415.
URL http://doi.acm.org/10.1145/1102351.1102415

[114] R. K. Ando, T. Zhang, A high-performance semi-supervised learning method for text chunking, in: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL '05, Association for Computational Linguistics, Stroudsburg, PA, USA, 2005, pp. 1–9. doi:10.3115/1219840.1219841.
URL https://doi.org/10.3115/1219840.1219841

[115] J. Blitzer, R. McDonald, F. Pereira, Domain adaptation with structural correspondence learning, in: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing,

EMNLP '06, Association for Computational Linguistics, Stroudsburg, PA, USA, 2006, pp. 120–128.
URL `http://dl.acm.org/citation.cfm?id=1610075.1610094`

[116] A. Argyriou, T. Evgeniou, M. Pontil, Convex multi-task feature learning, Machine Learning 73 (3) (2008) 243–272. doi:10.1007/s10994-007-5040-8.
URL `http://dx.doi.org/10.1007/s10994-007-5040-8`

[117] A. Schwaighofer, V. Tresp, K. Yu, Learning gaussian process kernels via hierarchical bayes, in: L. K. Saul, Y. Weiss, L. Bottou (Eds.), Advances in Neural Information Processing Systems 17, MIT Press, 2005, pp. 1209–1216.
URL `http://papers.nips.cc/paper/2595-learning-gaussian-process-kernels-via-hierarchica`
`pdf`

[118] N. D. Lawrence, J. C. Platt, Learning to learn with the informative vector machine, in: Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, ACM, New York, NY, USA, 2004, pp. 65–. doi:10.1145/1015330.1015382.
URL `http://doi.acm.org/10.1145/1015330.1015382`

[119] L. Mihalkova, et al., Transfer learning from minimal target data by mapping across relational domains (2009).

[120] L. Mihalkova, T. Huynh, R. J. Mooney, Mapping and revising markov logic networks for transfer learning, in: Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1, AAAI'07, AAAI Press, 2007, pp. 608–614.
URL `http://dl.acm.org/citation.cfm?id=1619645.1619743`

[121] M. Richardson, P. Domingos, Markov logic networks, Machine Learning 62 (1) (2006) 107–136. doi:10.1007/s10994-006-5833-1.
URL `http://dx.doi.org/10.1007/s10994-006-5833-1`