University of Miskolc

Faculty of Mechanical Engineering and Informatics

# Tools for Extracting and Forecasting Detailed Consumption Patterns From Cloud Traces

PhD Dissertation

Author: **Shallaw Mohammed Ali**
MSc in Software Engineering

Jozsef Hatvany Doctoral School of
Information Science, Engineering and Technology

Head of Doctoral School: **Prof. Dr. Jeno SZIGETI**
Academic Supervisor: **Prof. Dr. Gabor Kecskemeti**
Co-Supervisor: **Dr. Karoly Nehez**

Miskolc
2025

# Declaration

The author hereby declares that this thesis has not been submitted, either in the same or in a different form, to this or to any other university for obtaining a PhD degree. The author confirms that the submitted work is his own and the appropriate credit has been given where reference has been addressed to the work of others.

**Shallaw Mohammed Ali**
**2024**

# Acknowledgments

First and foremost, I express my utmost gratitude to Almighty Allah for His unending mercy, guidance, and blessings, without which this achievement would not have been possible.

I am sincerely grateful to **Dr. Gábor Kecskeméti**, my supervisor, for his outstanding academic mentorship, insightful research guidance, and continuous support throughout my studies. His expertise and dedication have been instrumental in shaping this work.

I also extend my heartfelt appreciation to the Stipendium Hungaricum Scholarship Programme, the University of Miskolc, and the Iraqi government for their generous support and for providing me with the opportunity and resources to pursue this academic journey.

I would like to express my profound gratitude to **my parents and siblings** for their unconditional love, prayers, and steadfast belief in me. Their constant support has been the foundation of all my accomplishments.

To **my beloved wife**, thank you for your patience, love, and unwavering support. You have been my source of serenity and strength, offering comfort and motivation throughout this journey.

Lastly, I would like to thank my **friends**, whose kindness and companionship have made this experience all the more rewarding.

**Shallaw Mohammed Ali**

# Contents

# List of Tables

# List of Figures

## List of Abbreviations

| | | |
|---|---|---|
| **DB** | Davies-Bouldin | 19 |
| **SC** | Silhouette-Coefficient | 21 |
| **CH** | Clainski-Harabasz | 21 |
| **SeQual** | Sequential method of clustering Quality | 49 |
| **EFection** | Effectiveness detection of clustering quality | 50 |
| **MICRAST** | Micro-forecasting approach for cloud user consumption pattern based on RNN | 50 |
| **RNN** | Recurrent Neural Network | 24 |
| **LSTM** | Long Short-Term Memory | 25 |
| **GRU** | Gated Recurrent Unit | 25 |
| **MAPE** | Mean Absolute Percentage Error | 28 |
| **$R^2$** | Coefficient of Determination | 13 |
| **LS** | Laplacian Score | 22 |
| **RF** | Relief Filter | 22 |
| **PCF** | Pearson Correlation Filter | 22 |

# Chapter 1

# Introduction

Data is one of the most important assets for studying different aspects of applied computing. It provides the foundation for understanding complex phenomena and uncovering valuable insights. For example, datasets capturing system dynamics and stability, such as those used in [1] and [2], offer crucial input for modelling and interpretation. In cloud computing, workload traces have been widely used as rich data sources for supporting more efficient resource management, as shown in [3] and [4].

To gain the most benefit from these data, clustering and forecasting methods have been widely exploited as key techniques for data analysis [5, 6]. Clustering helps to extract important information from cloud traces, such as detailed patterns of resource consumption in users' records [7]. These patterns are necessary for implementing crucial resource management tasks (e.g., resource provisioning). It enables a more insightful understanding and, consequently, better forecasting of future consumption. Therefore, developing tools for such extraction and forecasting will ensure the identification of potential challenges and opportunities associated with resource utilisation and management.

These tools benefit various studies. For instance, Zhou et al. [3] introduced an energy consumption model for data centers. This model comprehensively considers both the energy usage of various hardware components (CPU, memory, disk, NIC) and the specific demands of different application types (CPU-intensive, transactional web, I/O-intensive). This work mainly focuses on application characteristics and hardware subsystems. However, targeting only the hardware aspect without considering the human factor wouldn't provide sufficient energy analysis and management [8] [9]. Efficient extraction and forecasting of consumption patterns at a detailed level would enable the model to move beyond application-centric generalisations and embrace a more nuanced, user-centric approach. This shift allows the model to achieve more accurate energy consumption predictions and facilitate dynamic resource allocation tailored to specific user needs.

In [10], Chen et al. presented StressCloud, a tool designed to analyse the performance and energy consumption of cloud systems. The researchers used StressCloud to conduct experiments, profiling how different workloads and resource allocation strategies impact both performance and energy use in a controlled cloud environment. However, the authors neglected to incorporate consumption pattern analysis into StressCloud, which could significantly enhance its capabilities. It would enable the creation of dynamic, user-driven workloads for more realistic performance and energy consumption analysis

In addition, Kecskemeti et al. [4] demonstrated that users' behaviours have a great effect on maintaining the free and unconstrained availability of cloud resources. Therefore, they proposed offering these users virtual tokens (so-called engaging options) to improve resource efficiency. This mechanism would require analysis of users' patterns at a detailed level to target the engaging options more efficiently towards the desired behaviour. The analysis process for these studies needs to be unsupervised, as many cloud records show ambiguity in their users' labelling.

## 1.1 Research Problems

Much of the literature has provided tools for trace analysis based on extraction via clustering and forecasting such as [7] and [11]. However, these works have overlooked supporting such tools with a detailed consideration of the human aspect. Analysing human patterns at this level is crucial for efficient resource and energy management [9], [8]. In this context, clustering-based studies in cloud computing have neglected two of the main factors that affect clustering quality: the selection of dimensions (attributes) and the methods of clustering [12], [13]. Both must be carefully considered to ensure that the clustering is not only accurate but also useful for subsequent analysis or decision-making processes. This is particularly pertinent in cloud computing, where workloads are often characterised by large-scale, dynamic, and complex datasets.

Regarding attribute selection, many data analysis studies, such as [14] and [15], have exploited methods of feature selection and dimensionality reduction. According to [16], [17], and [18], the use of such general feature selection methods requires supervisory inputs (e.g., predefined labels and categories). These inputs are not typically available in cloud workload traces. Meanwhile, to address clustering method selection, studies often rely on generic and unautomated techniques, which are not reliable for repeated tasks. In addition, these tools require full trace analysis. This is not applicable to cloud traces, since many of these traces contain millions of lines of user records, making the full analysis process costly and less efficient for detailed-level pattern extraction.

On the other hand, for forecasting, related research such as [19], [20], and [21] has presented various prediction models. However, the forecasting approach in these studies was designed to deal with consumption patterns as trends by performing predictions at an overall level, which we will refer to as macro-prediction in this dissertation. Thus, such an approach lacks the ability to capture users' hidden patterns from cloud traces and to forecast them at a detailed, micro-prediction level.

## 1.2 Research Aims

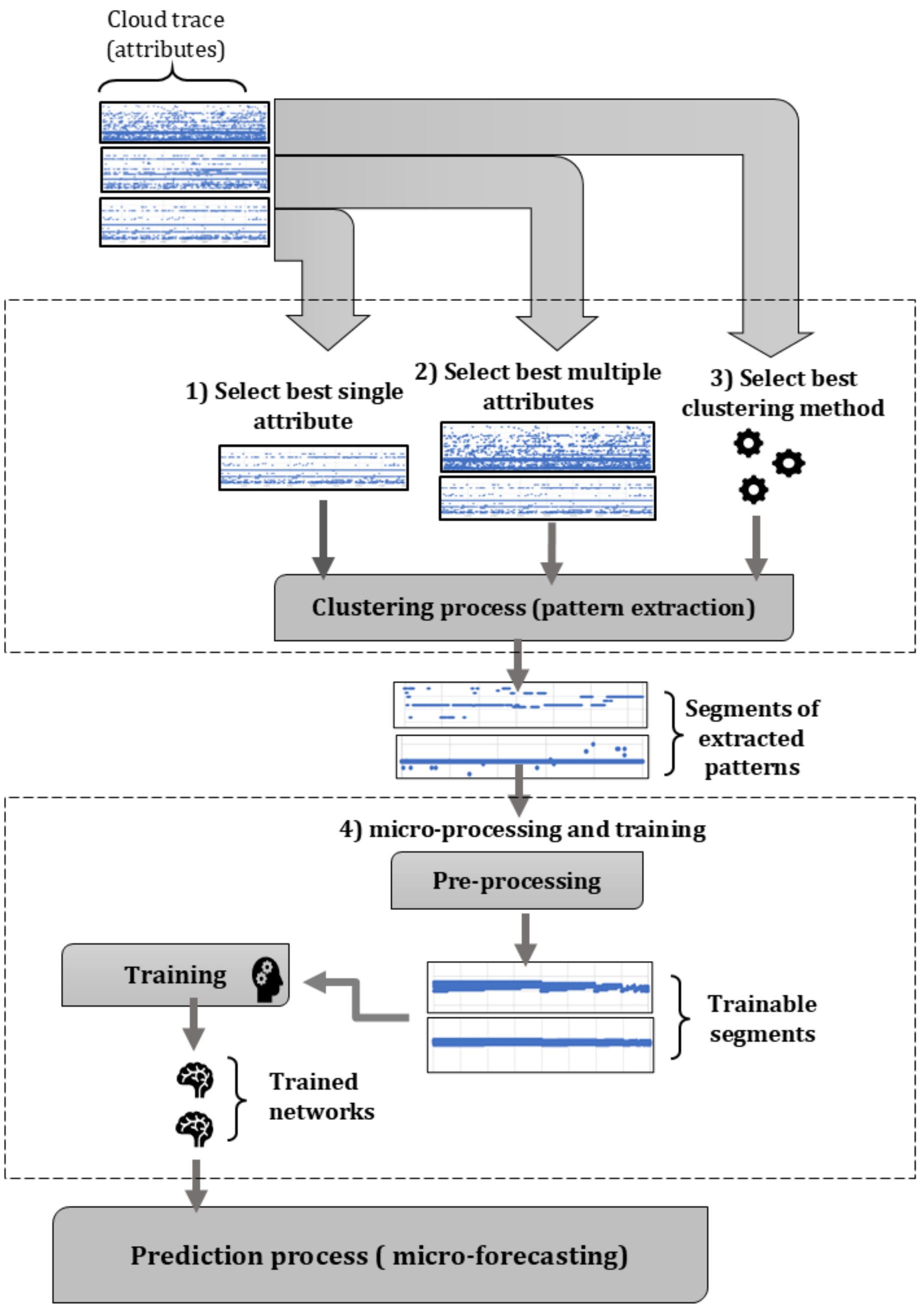
This thesis aims to address the following gaps in existing analysis tools:

- Attribute and method selection that enable the extraction of patterns from cloud traces at a detailed level.
- A micro-prediction approach capable of capturing and accurately predicting these detailed patterns.

To address these gaps, our research is divided into the goals shown in Figure 1.1. These goals are outlined as follows:

- The study of tools for attribute and clustering method selection that are more efficient for the characteristics of cloud traces. These tools should perform such selection for both single and multiple attributes, as well as clustering methods, without the need for predetermined parameters. They should be automated and unsupervised, and should not require the analysis of full traces. These tools should enable clustering methods to produce segments of extracted detailed patterns.
- The study of forecasting approaches that support capturing and predicting detailed patterns from cloud traces. This approach needs to integrate the extraction tools described above and apply separate preprocessing to produce trainable segments for each of the extracted patterns. This needs to be followed by separate training for each of these segments to generate a trained network for each. This enables more accurate micro-forecasting.

Accordingly, the overall goal of this dissertation is *to develop automated, unsupervised tools for extracting and forecasting detailed cloud patterns without requiring full-trace analysis or predefined parameters.*



Cloud trace (attributes)
1) Select best single attribute
2) Select best multiple attributes
3) Select best clustering method
Clustering process (pattern extraction)
Segments of extracted patterns
4) micro-processing and training
Pre-processing
Training
Trainable segments
Trained networks
Prediction process ( micro-forecasting)

Figure 1.1: Outline of research aims

## 1.3 Dissertation Guide

The rest of this dissertation is structured as follows: Chapter 2 covers the background of user behaviour analysis tools, including both data extraction and forecasting aspects. It also presents a literature review and related works regarding the implementation of these two aspects in cloud computing and their limitations. Next, Chapter 3 describes the methodology adopted in this thesis for developing the tools shown in Figure 1.1, with the aim of enabling more efficient analysis to address existing gaps. This involves proposing extraction tools for attribute and method detection, along with a new forecasting approach. Subsequently, Chapter 4 presents the evaluation process of the proposed tools and the experimental results for each. Finally, Chapter 5 concludes the thesis and outlines the contributions and future directions.

# Chapter 2

# Background and Literature Review

This chapter provides a comprehensive overview of the key background and foundational elements that support this thesis. Additionally, it reviews the existing literature, highlighting major findings and identifying the gaps that this dissertation aims to address in Section 1.2.

## 2.1 Background

In this section, we present the fundamental knowledge necessary to support the rest of this dissertation. We start by outlining the characteristics of typical cloud traces, as understanding these traces is essential for thorough analysis. Then, we describe the investigation and development environment used for data extraction and forecasting.

### 2.1.1 Typical Cloud Workload Traces

Cloud traces are datasets that record logs of users' interactions with cloud services and other computing environments [22]. These logs include users' requests and resource consumption on these services. Cloud traces can be used for various purposes, such as data analysis, resource management, and prediction. They are available through resources such as the Grid Workload Archive (GWA) [23] and the Logs of Real Parallel Workloads from Production Systems (PWA) [24].

Typically, the traces from these resources are stored in structured formats. All these formats organise users' logs as fields (attributes). Each attribute stores specific information about each user, such as requested time, run time, etc. The GWA stores users' logs in Grid Workload Format (GWF), and the PWA stores logs in Standard Workload Format (SWF) [25]. By checking their two formats, it is inferred that both GWF and SWF share similar attributes, except for minor differences. Table 2.1 shows the characteristics of the attributes that describe cloud users' traces in these two formats. The table also shows that both formats have attributes for user labels. The providers of the datasets can elect to omit some attributes. Thus, we consider only those traces where user labels are consistently offered. We refer to these traces as *supervised traces*.

Table 2.1: Attributes in the GWF and SWF formats

| **Attributes (GWF)** | **Attributes (SWF)** | **Information** |
|---|---|---|
| Job ID | Job number | A counter field, starting from 1. |
| Submit Time | Submit Time | In seconds. The submittal time of the first job. |
| Wait Time | Wait Time | In seconds. The difference between the job's submit time and the time at which it actually began to run. |
| Run Time | Run Time | In seconds. The wall clock time the job was running. |
| Nprocs | No. of allocated Process | An integer. The number of processors the job uses. |
| Avg. CPU Time Used | N/A | Both user and system, in seconds. This is the average CPU time used. |
| Used Memory | Used Memory | In kilobytes. This is again the average per processor. |
| Req. NProcs | Req. Number of Processors | An integer. Refers to the number of processors required. |
| Req. Time | Req. Time | Measured in wallclock seconds. This could represent runtime or average CPU time per processor, determined by a header comment. |
| Req. Memory | Req. Memory | Kilobytes per processor. This is the size of the memory requested by the user. |
| User ID | User ID | (GWF: a string, SWF: a natural number.) This is the label of an individual user. |

*Continued on next page*

| **Attributes (GWF)** | **Attributes (SWF)** | **Information** |
|---|---|---|
| Group ID | Group ID | (GWF: a string, SWF: a natural number.) Some systems control resource usage by groups rather than by individual users. |
| Ex. ID | Ex. App Number | (GWF: a string, SWF: a natural number.) In some logs, this might represent a script file used to run jobs rather than the executable directly. |
| Queue ID | Queue Number | (GWF: a string, SWF: a natural number.) The number of different queues in the system. |
| Part ID | Partition Number | (GWF: a string, SWF: a natural number.) The number of different partitions in the system. |
| Orig Site ID | N/A | A string. Used to categorize the site from which the job originated. |
| Last Run Site ID | N/A | A string. Used to describe the last site in which the job ran. |
| Used Network | N/A | In kilobytes/s. This is again the average per processor. |
| Used Resources | N/A | List of comma-separated Resource Description: Consumption. |
| Req. Platform | N/A | CPU architecture, OS version. |
| Req. Resources | N/A | Refers to other requested resources. |
| Project ID | N/A | The project ID, as a string (offers potential accounting details). |

While some traces in the above archives do not follow their respective formats, such as BitBrain and Materna, a new format is used for these traces, whose attributes are shown in Table 2.2. These traces provide a much more detailed summary. For example, CPU time averages can be calculated based on provisioned CPU capacity and utilisation metrics, whereas such details are not disclosed in the other traces. According to the table, these traces do not provide any attribute that directly indicates the identification of cloud users. We refer to these traces as *unsupervised traces.*

Other types of cloud traces are those with less structured formats (e.g., web applications, serverless cloud functions, IoT systems, or platform-specific services like Azure). Unlike the standardised traces discussed above, these

Table 2.2: Attributes in the BitBrain and Materna Datasets

| **Attributes** | **Information** |
|---|---|
| Timestamp | Number of milliseconds since 1970-01-01. |
| CPU cores | Number of virtual CPU cores provisioned. |
| CPU provisioned (requested) | The capacity of the CPUs in terms of MHz. |
| CPU usage (MHZ) | Usage in terms of MHz. |
| CPU usage (percentage) | Usage in terms of percentage. |
| Memory provisioned (requested) | The capacity of the memory of the VM in terms of KB. |
| Memory usage | The memory that is actively used in terms of KB. |
| Memory usage (percentage) | In terms of (only in GWA-T13-materna-trace). |
| Disk read throughput | In terms of KB/s (only in GWA-T-12 Bitbrains). |
| Disk write throughput | In terms of KB/s. |
| Disk size | In terms of GB (total sum of all virtual HDDs) (only in GWA-T13-materna-trace). |
| Network received throughput | In terms of KB/s. |
| Network transmitted throughput | In terms of KB/s. |

traces may lack a consistent structure or schema, making them more variable and potentially leading to unreliable results. Therefore, we disregarded these traces in this dissertation. Table 2.3 provides examples of common attributes found in such traces.

Table 2.3: Typical characteristics (attributes) of less structured trace formats

| **Attributes** | **Information** |
|---|---|
| Timestamp | When the event/log/trace occurred (could be inconsistent in format, e.g., ISO, epoch). |
| Service/Component Name | Identifies the origin of the trace (e.g., API Gateway, Azure Function, IoT Hub). |
| Log Level (if any) | Severity like INFO, ERROR, DEBUG—not always standardized. |
| Request ID / Trace ID | May or may not be present; useful for linking distributed operations. |
| Operation Name | Describes the action (e.g., getUserData, deviceHeartbeat). |
| User / Session ID | May include user ID, session token, or user agent—varies widely. |
| Latency / Duration | time taken by an operation; often logged in custom ways. |
| Environment Metadata | May include region, instance ID, or function version (especially in Azure/AWS). |
| Requested Resources | Indicates requested or provisioned resources (e.g., CPU, memory, time limits). Present in HPC jobs as explicit requests; in cloud traces, may appear as limits, instance types, or resource usage hints. |

### 2.1.2 Extraction Environment

One important task of the proposed tools is the ability to extract detailed information about consumption patterns from cloud traces. Such extraction can be achieved using clustering methods. Efficient clustering requires tools for detecting the best among the given attributes and methods for clustering the targeted traces. In this subsection, we provide an overview of each of these tools, along with a thorough description of clustering and its validation metrics.

### Clustering

Clustering is the process of grouping similar objects into clusters [26]. It is widely used as one of the main data analysis methods, especially for revealing crucial information from datasets. Therefore, this thesis studies clustering methods as a vital tool for extracting detailed patterns from cloud traces.

There are three main types of clustering methods: Partitioning, Hierarchical, and Fuzzy [27]. In partitioning methods (which we mainly use in this thesis), clustering is conducted by grouping similar data points together while maximising the differences between the clusters. Examples of these types of methods are K-means, Cascade Simple K-means, and Gaussian Mixture Models (GMM). In hierarchical methods, data points are organised into nested clusters based on their similarities or distances, creating a tree-like structure known as a dendrogram to illustrate the relationships between clusters [28].

While fuzzy methods, or soft clustering, enable data points to be part of multiple clusters with different levels of membership. This is unlike traditional hard clustering, where each data point is strictly assigned to a single cluster [29]. Nevertheless, fuzzy methods suffer from that their performance can be sensitive to the choice of parameters, such as the fuzziness parameter (m in Fuzzy C-Means). These types may not be suitable for data with few parameters, such as cloud traces. Partitioning methods are the simplest. They are suited for numeric datasets and require short computation run times [30], whereas others are more complex and require more time to run. To achieve efficient clustering quality, there are factors that need to be considered carefully, such as which attributes in a given trace and which method should be selected for clustering. In addition, effective validation of the clustering process is vital for better analysis.

### Data Analysis and Selection

As outlined in Section 2.1.1, the cloud traces from different resource archives are typically categorised into two types: supervised and unsupervised. For efficient extraction implementation, we carried out the following analysis tasks on both types of cloud traces.

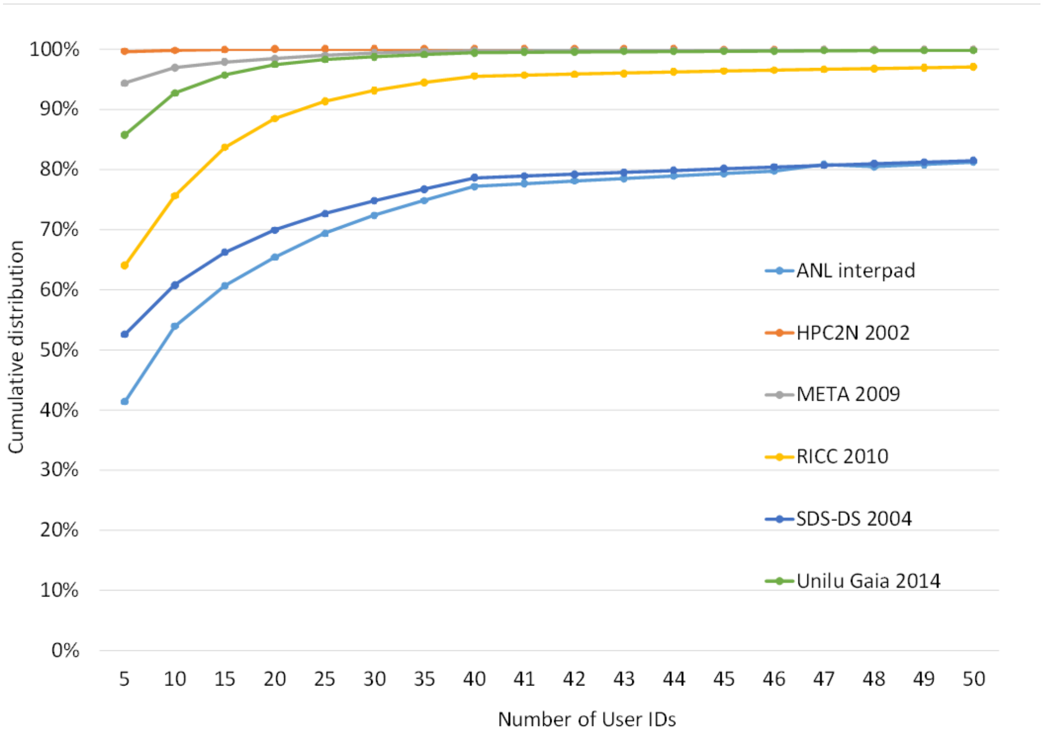


Figure 2.1: Sample cumulative percentage distribution of consumption patterns across five cloud traces

- First, we analysed the traces in Table 2.1 and 2.2 for their usability for clustering. We classified the attributes of each trace into three groups for supervised traces and two groups for unsupervised ones, regarding their usefulness for clustering. The first group consists of attributes with fewer than three unique values. Using such attributes as input to clustering could mislead the clustering algorithm, even if these attributes are combined with others. The second group (in the supervised traces) includes attributes with fewer unique values than the number of unique user IDs in the traces. These attributes are more likely to be combined with other attributes to reveal users' labels through clustering. The last group consists of attributes that have a wide variety of values and thus are more applicable for extraction.

- Since efficient clustering requires a thorough analysis of the data distribution, we measured the cumulative distribution of users' records for all the supervised traces in Table 2.1. In this context, the cumulative distribution reveals the actual number of users that represent the majority of records. It was observed that most of these traces exhibited a cumulative distribution ranging from 81% to 99% for only 50 unique user IDs, as shown in the samples in Figure 2.1. This suggests the potential for clustering traces into approximately 50 dominant behavioural groups, driven by the most active user IDs. This also highlights the necessity of further pattern extraction to validate and characterise these behavioural groups.

- To check which attributes are most likely better for combination, we measured the correlation between all the attributes of each trace in Table 2.1 using the Coefficient of Determination ($R^2$) metric. Table 2.6 and Table 2.7 represent samples of the $R^2$ values between each pair of the trace's attributes. These samples show the potential for correlations above 0.5 between cloud trace attributes, indicating a greater potential for effective attribute combination in clustering.

Table 2.4: Description of supervised traces

| Trace Name | attributes with unique Values < 3 | attributes with unique Values < User IDs | Attributes applicable for clustering |
|---|---|---|---|
| ANL interpad log | Average CPU time used , Used memory, Requested Memory, Status, Group ID | Requested time, Requested Number of Processors | Submit Time, Run Time, Wait Time |
| PIKIPLEX | Used Memory, Queue ID , Requested Number of Processors | Requested Memory, Executable Number | Submit Time, Wait Time, Run Time, Number of Allocated Processors, Average CPU time used |
| KIT-FH2 | Avg. CPU time used, Used Memory, Req. Memory, Status, Executable ID, Queue ID, Think time | N/A | Submit Time, Wait Time, Run Time, No. Allocated Process, Req. Number of Process |
| CIEMAT Euler 2008 | Number of Allocated Process, Requested Number of Process, Queue ID | Status | Submit Time, Wait Time, Run Time, Average CPU time used, Used Memory |
| CTC-SP2 1995 | Used Memory, Req. Number of Processors, Req. Time, Req. Memory, Status, Queue ID | N/A | Submit Time, Wait Time, Run Time, No. Allocated Process, Avg. CPU time used |
| GWATDAS2 | Req. Memory | Queue ID, Status | Submit Time, Wait Time, Run Time, No. Allocated Process, Avg. CPU time used, Req. Number of Process, Req. Time |

**Table 2.4: Labelled Traces Description (continued)**

| Trace Name | Unique Values < 3 | Unique Values < User IDs | Applicable for Clustering |
|---|---|---|---|
| GWAT-LCG | Wait Time, Number of Allocated Process, Average CPU time used, Used Memory, Requested Number of Process, Requested Time, Status, Queue ID | N/A | Run Time, Submit Time |
| GWA-T-3 NorduGrid | Wait Time, Number of Allocated Process, Average CPU time used, Requested Time, Requested Memory, Status, Executable ID, Requested Number Process | Used Memory, Queue ID | Run time, Submit time |
| GWAT-10 SHARCNet | Req. Memory , Req. Time, Status, Executable ID, Req. Number of Process | Partition ID | Wait Time, Submit Time, Run Time, No. Allocated Process, Avg. CPU time used, Used Memory |
| UNILU-GAIA | N/A | Status, Requested Memory, Number of Allocated Processors | Submit Time, Wait Time, Run Time, Average CPU time used, Used Memory, Requested Number of Processors |
| CEACurie-2011 | Average CPU time used, Used memory, Requested Memory, Executable number, Queue ID | Status | Submit time, Wait time, Run time, Number of allocated process, Requested number of process |

**Table 2.4: Labelled Traces Description (continued)**

| Trace Name | Unique Values < 3 | Unique Values < User IDs | Applicable for Clustering |
|---|---|---|---|
| LANLO2K | Req. Number of Processors, Queue ID, Partition ID | Requested Memory | Submit Time, Wait Time, Run Time, Number of Allocated Process, Average CPU time used, Used Memory |
| SDS-Par 95 | Req. Number of Processors, Requested Time, Requested Memory, Executable Number | Queue ID | Submit Time, Wait Time, Run time, Number of Allocate Processors, Average CPU time used, Used Memory |
| MetaCentrum | Avg. CPU time used, Used Memory, Status | Queue ID, Requested Number of Processors, Number of Allocated Processors | Submit Time, Wait Time, Run Time, Requested Memory |
| LANLCM5 | N/A | No. Allocated Process, Partition ID, Requested Time | Submit Time, Wait Time, Run Time, Average CPU time used, Used Memory, Requested Number of Processors, Requested Memory, Status, Queue ID |
| RICC | Avg. CPU time used, Used Memory, Status , Queue ID | Requested Number of Processors, Requested Memory, Number of Allocated Processors | Submit Time, Wait Time, Run Time |

**Table 2.4: Labelled Traces Description (continued)**

| Trace Name | Unique Values < 3 | Unique Values < User IDs | Applicable for Clustering |
|---|---|---|---|
| SDSC-SP2 | N/A | Status, Partition number | Submit Time, Wait Time, Run Time, Number of Allocated Processors, Average CPU time used, Used Memory, Requested Memory, Requested Number of Processors, Requested Time |
| OSCClust-2000 | Requested Memory, Queue ID | Requested Number of Processors, Status | Submit Time, Wait Time, Run Time, Used Memory, Average CPU time used, Requested Time |
| GWA-T-4 AuverGrid | No. Allocated Processors, Requested Number of Processors, Status, Queue ID | Requested Memory, Req. Time | Submit Time, Wait Time, Run Time, Average CPU time used |

Table 2.5: Description of unsupervised traces

| Trace Name | Attributes with Unique Values < 3 | Attributes Applicable for Clustering |
|---|---|---|
| BitBrains | N/A | Timestamp, CPU Capacity provisioned [MHZ], CPU usage [MHZ], CPU usage [%], Memory Usage [KB], Disk read throughput [KB/s], Disk write throughput [KB/s], Network received throughput [KB/s], Network transmitted throughput [KB/s] |
| GWAT AuverGrid (-1 User ID) | Number of Allocated Processors, Average CPU time used, Used Memory, Requested Number of Processors, Requested Time, Status | Submit Time, Wait Time, Run Time, Requested Memory |
| Materna | Disk Size, CPU Core, CPU capacity provisioned [MHZ], Memory capacity provisioned [KB] | Time stamp, CPU usage [MHZ], CPU usage [%], Memory usage [KB], Memory usage [%], Disk read throughput [KB/s], Disk write throughput [KB/s], Network received throughput [KB/s], Network transmitted throughput [KB/s] |
| Facebook Hadoop Workload | N/A | Submit Time, inter.submit.gap.second, map.input.byte, shuffle.byte, reduce.output |
| Container usage (Ali Baba trace 2017) | N/A | Start time of measurement, online instance id, used percent of req. CPU (%), used percent of req. memory, used percent of req. disk space, linux CPU load average of 1 min, linux CPU load average of 5 min, linux CPU load average of 15 min, average cycles per instruction, average last level cache misses per 1000 instructions, maximum cycles per instruction, maximum last level cache misses per 1000 instructions |

Table 2.6: A correlation matrix of $R^2$ between the attributes of the CTC-SP2 1996 trace.

| Attributes | Submit time | Wait Time | Run Time | No. Alloc. proc. | Avg. cpu. | Req. No. Proc. | Req. Time | Req. Memo. |
|---|---|---|---|---|---|---|---|---|
| Submit Time | - | 0.093 | 0.39 | -0.173 | 0.37 | -0.26 | 0.53 | -0.75 |
| Wait Time | 0.09 | - | 0.13 | 0.67 | 0.11 | 0.46 | 0.16 | -0.11 |
| Run Time | 0.39 | 0.13 | - | -0.09 | 0.97 | -0.15 | 0.75 | -0.40 |
| No. Alloc. proc. | -0.17 | 0.67 | -0.09 | - | -0.09 | 0.699 | -0.129 | 0.130 |
| Avg. cpu. | 0.379 | 0.11 | 0.97 | -0.09 | - | -0.14 | 0.73 | -0.38 |
| Req. No. proc | -0.26 | 0.46 | -0.15 | 0.69 | -0.14 | - | -0.19 | 0.21 |
| Req. Time | 0.53 | 0.16 | 0.75 | -0.12 | 0.73 | -0.19 | - | -0.5 |
| Req. Memo. | -0.75 | -0.11 | -0.4 | 0.13 | -0.38 | 0.21 | -0.52 | - |

Based on the above, we found the traces in Tables 2.4 and 2.5 to be usable for later supervised and unsupervised implementations in this thesis. We selected these traces as inputs for the development of our tools in Section 3.2 and 3.3, and for the evaluation of these tools in Section 4.1 and 4.2. We also observed that each trace demonstrates different pairs of attributes with $R^2 > 0.5$. This means that there is no single pattern that applies to all attributes. In addition, according to our observation of the cumulative distribution, it is expected that the majority of detailed consumption patterns can be extracted within the range of 50 unique users (clusters).

### Clustering Validation

One of the fundamental aspects of data clustering is the assessment of its results through a process known as clustering validation [30]. Typically, clustering validation metrics are classified into two main types: internal and external [31]. Internal metrics measure the closeness and distances between the clusters, while external metrics usually depend on predefined classifications or ground truth. External validation metrics are typically used for supervised traces that provide expected labels for the data to be clustered, whereas internal validation metrics are used for unsupervised traces that do not disclose such information.

The most commonly used internal metrics are the Davies-Bouldin (DB),

Table 2.7: A correlation matrix of $R^2$ values between the attributes of the LANL CM5 trace.

| Attributes | Submit time | Wait Time | Run Time | No. Alloc. proc. | Avg. cpu. | Used Memo. | Req. No. Proc. | Req. Time | Req. Memo. |
|---|---|---|---|---|---|---|---|---|---|
| Submit Time | - | -0.01 | 0.34 | -0.10 | -0.15 | 0.54 | -0.16 | -0.15 | 0.57 |
| Wait Time | -0.01 | - | 0.12 | 0.51 | 0.21 | 0.03 | 0.36 | 0.24 | 0.08 |
| Run Time | 0.34 | 0.125 | - | -0.01 | 0.04 | 0.38 | -0.03 | 0.02 | 0.41 |
| No. Alloc. proc. | -0.1 | 0.51 | -0.01 | - | 0.37 | -0.058 | 0.69 | 0.5 | -0.08 |
| Avg. cpu. | -0.15 | 0.21 | 0.047 | 0.37 | - | -0.07 | 0.37 | 0.53 | -0.09 |
| Used Memo. | 0.54 | 0.038 | 0.381 | -0.058 | -0.07 | - | -0.09 | -0.064 | 0.75 |
| Req. No. proc | -0.161 | 0.36 | -0.03 | 0.699 | 0.37 | -0.09 | - | 0.64 | -0.12 |
| Req. Time | -0.15 | 0.24 | 0.02 | 0.5 | 0.53 | -0.06 | 0.64 | - | -0.09 |
| Req. Memo. | 0.57 | 0.08 | 0.41 | -0.08 | -0.09 | 0.75 | -0.129 | -0.09 | - |

Silhouette-Coefficient (SC), and Clainski-Harabasz (CH) [32]. The DB metric evaluates the clustering results by measuring the similarities and differences between the clustering groups [33]. The lower the metric score, the better the distances between these groups. In the SC metric, each clustering group is defined as a *silhouette*. The clustering quality is evaluated by measuring how well a data point fits within its own cluster compared to other clusters. Essentially, the idea of SC is that items in the same cluster should be close together, while items in different clusters should be farther apart. The scores for this metric range from -1 to 1, where 1 represents the highest quality and -1 the lowest. The CH metric measures clustering quality by comparing the variance between clusters to the variance within these clusters. In this metric, the higher the score, the better the quality.

In external validation, the *Entropy*, *Precision*, and the *Adjusted Rand Index* [34] are the most commonly used metrics. In Entropy, the quality is evaluated by measuring the homogeneity of the clustering results compared to the intended ones (i.e., labelled data in this dissertation) [35]. The Entropy score ranges from [0, 1], where 0 indicates the lowest quality and 1 indicates the highest. Precision evaluates the clustering quality by measuring the highest possible matches between the given ground truth and the clustering results [36]. Similarly, the results for this metric range between [0, 1]. In the *Adjusted Rand Index*, the quality of the clustering is measured by calculating the agreement between the clustering results and the given references [37]. The scores for the *Adjusted Rand Index* are bound between [-1, 1], where 1 represents perfect agreement between the clustering results and the ground truth, while -1 indicates complete disagreement. Typically, internal validation metrics (e.g., DB and SC) are used for developing analysis algorithms that require unsupervised assessment for the targeted datasets, whereas external validation metrics, such as *Precision*, are generally employed to evaluate the performance of these algorithms against a ground truth. Thus, we can exploit the use of these metrics in the development and evaluation of this dissertation's extraction tools.

### Dimensionality Selection Methods

Dimensionality selection methods are mainly aimed at reducing the number of dimensions in a dataset while retaining its essential structure and information. This process is crucial for ensuring clustering quality because high-dimensional data can lead to issues like the "curse of dimensionality," where the distance between points becomes less meaningful. In the context of the cloud traces, these dimensions are represented as attributes. Therefore, we will refer to these dimensions as attributes in this thesis.

The dimensionality selection methods can be categorised into two main types: dimensionality reduction and feature selection. *Feature selection* involves choosing attributes from the original traces based on their expected usefulness for clustering according to particular criteria [38,39]. On the other hand, *dimensionality reduction* aims to select the best combination of attributes by merging their useful characteristics into a single output. This process creates a new representation of the data that captures the essential information while reducing dimensionality.

Feature selection can be subcategorised into Filter, Wrapper, and Hybrid algorithms. In Filter algorithms, the optimal attributes are selected and ranked based on the general properties of the targeted trace. The main characteristic of these algorithms is that they require less execution time compared to others [17]. One commonly used filter algorithm is the Pearson Correlation Filter (PCF), in which all possible correlations between the attributes in the targeted trace are measured, and those with the highest dependency are disregarded. This is repeated for a user-defined number of parameters [18]. Another supervised algorithm is the Relief Filter (RF), which detects attributes that are statistically relevant [40]. The Laplacian Score (LS) algorithm is one of the most commonly used unsupervised types of filter methods. It ranks attributes based on their ability to maintain the similarity between data points, ensuring that the local structure of the data is preserved [41].

In wrapper methods, learning algorithms are combined with a clustering method and used in the evaluation process [39]. The selection process in these methods starts by ranking and detecting trace attributes using search strategies. Then, the ranked attributes are evaluated through a learning algorithm. Although these methods can show better ability to improve the intended learning algorithm, they depend on the algorithms used and tend to involve costly computational processes [17]. Hybrid methods combine the advantages of both filter and wrapper methods to design a more effective attribute selection algorithm [42]. According to [17], the limitation of hybrid methods is that filter and wrapper techniques may not be combined efficiently, which could cause low performance.

On the other hand, one of the well-known tools for dimensionality reduction is Principal Component Analysis (PCA). PCA aims to reduce dimensionality by converting a complex trace with many correlated attributes into fewer dimensions. It lowers these dimensions by creating new independent attributes derived from the original ones, known as principal components [43]. These components are ranked by the variance they capture from the data. Another tool to deal with dimensionality is the t-Distributed Stochastic Neighbor Embedding (t-SNE) technique. It is a machine learning algorithm used for visualising high-dimensional data by mapping it to a lower-dimensional space, typically two or three dimensions [44]. This is conducted by detecting the differences between data points, then minimising these differences in lower dimensions. Auto-encoders are also presented as a type of neural network-based dimensionality reduction algorithm. They are designed to capture efficient representations of the user-defined data through unsupervised learning, particularly useful for non-linear reduction [45]. In addition, Singular Value Decomposition (SVD) is developed to deal with dimensionality by decomposing the data matrix into three other matrices [46]. This technique is often used in natural language processing and image compression, but it can also be applied to reduce dimensionality in clustering tasks.

In conclusion, using general methods of feature selection and dimensionality reduction requires providing several parameters, such as the number of needed attributes and clusters. These parameters are not necessarily available in workload traces. Another disadvantage of dimensionality reduction tools is that they tend to generate new versions of the original attributes, which may affect extraction quality. Such approaches are not beneficial for pattern extraction since they cause changes in the trace's characteristics characteristics and information.

### 2.1.3 Forecasting Environment

The second aim of this dissertation is to provide a forecasting approach that captures and micro-predicts detailed patterns of users' consumption records, as highlighted in Section 1.2. Therefore, this subsection covers the essential knowledge about the concept of time series forecasting, its models, and the requirements for input data. It also includes the evaluation metrics for the forecasting process.

#### Time Series Forecasting and Models

In time series forecasting, prediction is performed based on data comprising a sequence of observations over time [47]. Two vital parameters of this prediction are the forecasting *window size* and *steps.* n this context, the window size represents the range of past events, which is a sequence of records in the trace that are utilised by the forecasting models. The number of future records to be predicted by these models is denoted by steps.

Forecasting models are typically categorised into two main types: statistical and ANN-based models. Statistical models, as the name indicates, use statistical techniques and assumptions about data distributions to reveal trends in historical records for predicting future variables. ANN-based models perform prediction using artificial neural networks to analyse and learn from past records. This section aims to cover models ranging from the simplest to the most advanced in these two categories, selected based on their range of usability.

Accordingly, Table 2.8 presents these models with their category and uses. We started with one of the very basic statistical forecasting methods, the Simple Moving Average (SMA) [48]. It estimates future data values by finding the mean of data points within a certain forecasting window [49]. SMA is best for short-term prediction of time series data with a stable trend. In the context of time forecasting, stability or stationarity means that the statistical properties (mean, variance, and auto-correlation) do not change over time. Another model is the Auto-Regression (AR) model, in which forecasting is performed through a linear combination of past values. The AR model is flexible for different types of time series patterns [50]. To form an Auto-regressive Moving Average (ARMA) model, AR is combined with another type of model, the Moving Average (MA), which uses past errors to predict future events in a regression-like model [50]. In ARMA, the AR part predicts the current event based on past values, while the MA part calculates the errors of past predictions to correct the current one. ARMA is suitable for stable series with no trend or seasonality. From this mix, Auto-regressive Integrated Moving Average (ARIMA) was introduced by Box and Jenkins by adding integrated differencing to ARMA to convert the targeted data to stability [51]. This makes ARIMA usable for non-stable time series as well as for both short- and long-term forecasting. However, it cannot detect non-linear characteristics in the data, such as abrupt changes or variable interactions [50].

It is important to note that the above-described models are applicable only for uni-attribute forecasting, as depicted in Table 2.8. Thus, the Vector Auto-Regression (VAR) model was introduced as the multi-attribute version of the statistical model, used for predicting multiple attributes. In VAR, the next value for each attribute is predicted based on its own previous history in addition to the histories of other related attributes [52]. In the context of cloud traces, related attributes represent the consumption records of users in the same trace.

On the other hand, among the ANN-based forecasting models, this section covers the following: RNN, LSTM, and GRU, which consist of closed loops of network connections and feedback. These networks are developed to learn sequential patterns in time series data [53]. Recurrent Neural Network (RNN) is useful for stable time series data. However, according to Bengio et al. [54], one of the limitations of RNN is the challenge of vanishing

Table 2.8: The discussed forecasting models

| **Model** | **For** | **Category** |
|---|---|---|
| SMA<br>AR<br>ARMA<br>ARIMA | Uni-attribute | Statistical |
| VAR | Multi-attributes | |
| LSTM<br>GRU<br>RNN | Both | ANN-based |

gradients when the forecasting window increases. These gradients are used to update the network's weights during training. This causes the network to struggle to learn from earlier time steps, making it hard to capture long-term dependencies in the trace.

To overcome this challenge, the literature introduced the concept of Long Short-Term Memory (LSTM). LSTM accomplishes this by discarding irrelevant information in the network using gating mechanisms, which enable it to deal with long-term forecasting windows [55]. Cho et al. [56] proposed an improved version of RNN with gate optimisation called Gated Recurrent Unit (GRU). GRU has a similar structure to that of LSTM and is also used to address the issue of vanishing gradients in time series forecasting. It is worth mentioning that an essential advantage of ANN-based models is that they can be employed for both multi-attribute and uni-attribute forecasting scenarios. Table 2.8 presents these models and their uses.

### Data Analysis and Selection

In the forecasting area, the majority of prediction models are based on the assumption that the data of interest is stable [57]. Such stability indicates that the statistical properties of the data do not change over time, which makes it simpler to analyse the prediction process. Accordingly, the cloud traces need to be analysed for stability to ensure efficient forecasting. Without meeting the stability condition, forecasting results may turn out to be unreliable. Typically, to check the stability of the targeted traces, unit root tests are used. To perform the unit root test, several methods are employed. Among these are the Augmented Dickey-Fuller (ADF), Phillips-Perron (PP), and Zivot-Andrews tests [58]. According to [59], one of the most commonly used methods is ADF. It tests the data according to the following two hypotheses:

- Null hypothesis: the dataset has a unit root, and thus it's non-stable.
- Alternate hypothesis: the dataset doesn't have a unit root, and it's stable.

Therefore, we checked the stability of the cloud traces from the resources of the Grid Workload Archive and the Parallel Workload Archive to ensure efficient forecasting. To this end, we employed the ADF test for its high efficiency, as it is the most commonly used test in the related literature. We applied this test to users' patterns for the Requested Number of Processors across all the traces in Table 2.4, as it reflects their consumption records. Then, we calculated the average of the ADF results for the corresponding trace.

The results showed a p-value below 0.05, which represents the stability threshold for all these traces, with ADF statistic values shown in Table 2.9. These values ranged from –3.5 to –20 for all the supervised traces (and only Bitbrain in the unsupervised traces). This means that all the traces in Table 2.4 fall below the standard critical values commonly used in the literature (see Table 2.9). Since the p-values for each cloud trace were below 0.05, the null hypothesis is rejected, indicating that these traces are stable. Nevertheless, the strength of stability is not equal across all traces. The farther the trace's statistic is from the critical value, the stronger its stability. For instance, CTC SP2, with a statistic of –20, can be considered to have very strong stability. In contrast, UNILU Gaia, which recorded the lowest statistic value of –3.5, has the weakest stability among these traces and requires more careful processing during forecasting.

Despite exhibiting high stability, many of these traces, such as ANL-Interpad, SDSC Par 1995, OSC Cluster, and CEA Curie, showed non-linearity with abrupt changes. They also recorded a high standard deviation (SD). This is evident when their scales are examined, as shown in the example in Figure 2.2. These findings suggest that cloud workload traces may exhibit irregular changes without following a seasonal pattern, yet still maintain a degree of statistical stability. Such characteristics require pre-processing to reveal meaningful patterns and trends from these traces, making them

Table 2.9: ADF critical values for stability testing

| Level of Significance | Critical Value |
|---|---|
| 1% | -3.43 |
| 5% | -2.862 |
| 10% | -2.567 |

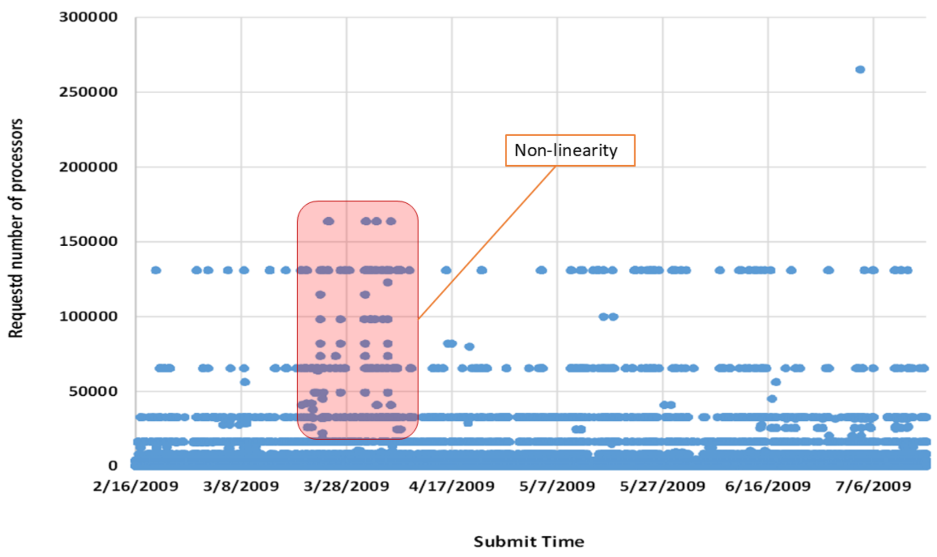


Figure 2.2: The characteristics of the ANL-Interpad trace

more suitable for prediction model training. Furthermore, since our analysis tools aim to provide a micro-prediction approach, it is vital to evaluate this approach using cloud traces that are suitable for such a purpose. To be applicable, these traces need to meet the following criteria:

- The trace should provide attributes that represent users' logs in numerical format. Such a format makes it possible to extract these log patterns and enables the forecasting model to capture them more efficiently.

- The trace should include job submission times for each user. This enables the formation of a sequence of events for each user based on their job timings. These sequences are essential for allowing forecasting models to learn from past events.

- The size of the trace should be sufficiently large to enable effective learning. Based on previous observations, ANNs struggle to learn from trace time series with fewer than 25K data points. Therefore, a minimum trace size of 25K data points is expected for a trace to be considered suitable.

- The trace should demonstrate a sense of stability, as most forecasting models assume that the characteristics of the targeted dataset are stable, as illustrated in Section 2.1.3, see page 25.

Based on the above, the traces listed in Table 2.12 were selected for meeting the specified criteria. This table presents the trace name, its size (in number of lines), the time period covered, and the corresponding ADF test results.

It is vital to emphasise that another prerequisite for achieving efficient forecasting is that the data should be uniformly sampled. This is particularly important when the information in the dataset is provided on different time scales. But what should be done when the dataset is not collected in a uniformly sampled manner? Such uniformity can be achieved through the implementation of time alignment and linear interpolation methods. Time alignment ensures that the action points in the data are synchronised with the corresponding time steps they represent [60]. Linear interpolation, on the other hand, fills in the gaps where data is missing. Essentially, it joins two known values with a straight line and approximates the intermediate values [61].

To demonstrate this, two samples of time series data from the ANL-Interpad trace are provided. Table 2.10 shows the trace structure before applying the uniforming process, where unaligned time steps and user IDs are clearly visible. Table 2.11, by contrast, shows the same trace after applying time alignment and linear interpolation methods, resulting in a more uniform structure. It is also evident that the Submit Time series becomes more aligned with ascending User IDs, making it more suitable for forecasting model training. This illustrates the necessity of applying a uniforming process to cloud trace data.

### Forecasting Validation

Forecasting validation is the process of measuring the efficiency of the employed model in predicting future events. It is typically conducted by comparing the predicted outcomes with the actual ground truth. In the context of cloud traces, not all datasets are suitable for training and using a portion as ground truth, as some lack a sufficient amount of usable records for this purpose. Forecasting validation is primarily implemented to assess whether the model is accurate enough during the forecasting testing process.

Four of the most well-known validation metrics are Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), and $R^2$ [62]. MAE is a statistical metric that evaluates the overall accuracy of a regression model by averaging the absolute differences between predicted and actual values. In contrast, MAPE calculates the average absolute percentage error, providing a relative measure of prediction accuracy [63]. It presents results on a percentage scale from 0 to $+\infty$ (where

Table 2.10: Time series sample of the ANL-Interpad trace before uniforming

| Submit Time | Requested Number of processors | User ID |
|---|---|---|
| 2009-01-01 00:00:00 | 2048 | 1 |
| 2009-01-01 00:00:07 | 2048 | 1 |
| 2009-01-01 00:26:30 | 2048 | 1 |
| 2009-01-01 00:36:45 | 8192 | 2 |
| 2009-01-01 00:42:46 | 2048 | 1 |
| 2009-01-01 00:45:51 | 64 | 3 |
| 2009-01-01 01:31:25 | 16384 | 4 |
| 2009-01-01 01:49:13 | 64 | 3 |
| 2009-01-01 02:52:35 | 64 | 3 |
| 2009-01-01 03:55:58 | 64 | 3 |
| 2009-01-01 03:58:33 | 2048 | 1 |
| 2009-01-01 06:05:41 | 2048 | 1 |
| 2009-01-01 07:22:26 | 2048 | 1 |
| 2009-01-01 07:38:41 | 2048 | 1 |

Table 2.11: Time series sample of the ANL-Interpad trace after uniforming

| Submit Time | Requested Number of processors | User ID |
|---|---|---|
| 2009-01-01 00:00:00 | 2048 | 1 |
| 2009-01-01 01:00:00 | 2046 | 1 |
| 2009-01-01 02:00:00 | 2044 | 1 |
| 2009-01-01 03:00:00 | 2042 | 1 |
| 2009-01-01 04:00:00 | 2040 | 1 |
| 2009-01-01 05:00:00 | 2038 | 1 |
| 2009-01-01 06:00:00 | 2036 | 1 |
| 2009-01-01 07:00:00 | 2034 | 1 |
| 2009-01-01 08:00:00 | 2032 | 1 |
| 2009-01-01 09:00:00 | 2030 | 1 |
| 2009-01-01 10:00:00 | 2028 | 2 |
| 2009-01-01 11:00:00 | 2026 | 2 |
| 2009-01-01 12:00:00 | 2024 | 2 |
| 2009-01-01 13:00:00 | 2022 | 2 |

Table 2.12: Cloud workload traces selected for forecasting investigation

| Trace | Trace size | Time period | ADF statistics |
|---|---|---|---|
| KTH-SP2-1996 | ≈30K | Sep-1996 to Aug-1997 | -5.3 |
| UNILU Gaia | ≈50K | May-2014 to Aug-2014 | -3.5 |
| ANL-interpad | ≈70K | Jan-2009 to Sep-2009 | -13.7 |
| SDSC-SP2-1998 | ≈75K | Apr-1998 to Apr-2000 | -4.9 |
| CTC-SP2-1996 | ≈8K | Jun-1996 to May-1997 | -20.2 |
| KIT-FH2-2016 | ≈100K | Jun-2016 to Jan-2018 | -6 |
| META CENTRUM-2009 | | Dec-2008 to Jun-2009 | -11.6 |
| LLNL Thunder-2007 | | Jan-2007 to Jun-2007 | -6.7 |
| LANL-O2K | | Nov-1999 to Apr-2000 | -7.5 |
| LANL CM5 1994 | ≈200K | Oct-1994 to Sep-1996 | -19 |
| HPC2N | | Jul-2002 to Jan-2006 | -12 |
| RICC-2010 | 400K | May-2010 to Sep-2010 | -12.11 |
| CEA Cuire-2011 | ≈700K | Feb-2011 to Oct-2012 | -10 |
| PIK-IPLEX | | Apr-2009 to Jul-2012 | -5.5 |
| SDSC-BLUE-2000 | ≈240K | Apr-2000 to Jan-2003 | -17 |
| LLNL-Atlas | ≈50K | Nov-2006 to Jun-2007 | -7 |
| Sandia ross 2011 | ≈60K | Nov-2011 to Jan-2005 | -6 |
| OSC Cluster | ≈80K | Apr-2000 to Nov-2001 | -4.4 |
| DAS2 | ≈200K | Jan-2003 to Jan-2004 | -5.6 |
| BitBrain | ≈1M | Oct-2012 to Feb-2013 | -7.9 |

0 is the best), making MAPE easier to interpret and widely used for forecasting evaluation [64]. RMSE measures how far off a model's predictions are from the actual values. Like MAE, RMSE reports errors in the same units as the data, without expressing their relative magnitude compared to the true values.

On the other hand, $R^2$ is a statistical measure of the degree of linear relationship between two data variables [65]. It ranks the relationship between the predicted and actual values. The $R^2$ ranges between 0 and 1, where values closer to 1 mean better.

Notably, both $R^2$ and MAPE provide a clear scale for measuring forecasting accuracy. These metrics accurately measure the degree of alignment between actual and predicted data. They also allow for clear and meaningful comparisons across different forecasting models. As discussed above, the accuracy measures for these metrics are presented as percentage-based values, while other metrics, such as RMSE, use actual values that may not be directly comparable. Based on these points, we employed MAPE and $R^2$ for the evaluation process of this thesis in Chapter 4.

## 2.2 In-Depth Review of Extraction in Cloud Computing

This subsection discusses previous works that studied the implementation of data extraction tools in cloud computing. It also highlights experimental review to investigate gaps and limitations found in these studies.

### 2.2.1 Related Works

While aiming to extract useful information from cloud workload traces, many studies have investigated the benefits of clustering methods. For example, Yousif et al. [7] proposed enhancing cloud resource utilization by clustering tasks in Google workload traces. This was done by clustering resource usage attributes such as CPU and hard disk usage using two clustering methods: K-means and density-based clustering. Similarly, Yu et al. [66] proposed a model to predict workload task patterns based on clustering CPU usage attributes from the same trace. Ghobaei-Arani [67] used the K-means method as a mechanism for cloud resource provisioning. This is conducted by categorising workloads based on their Quality of Service (QoS) requirements to enhance predictive accuracy and enable better resource allocation. Gao et al. [68] developed a hybrid approach incorporating K-means clustering to improve the accuracy of resource provisioning, also using Google cluster traces. Likewise, Shahidinejad et al. [69] used a hybrid clustering approach leveraging K-means to study and address resource provisioning issues.

Another application of clustering methods was conducted by [11]. In that paper, the researchers aimed to increase the accuracy of predicting parallel application runtimes to improve the utilization of parallel computing systems. This was carried out using datasets from a parallel workload for model training by targeting the application runtimes in these data without considering their specific users. Furthermore, Patel and Kushwaha [70] proposed exploiting two clustering methods, namely K-means and Gaussian Mixture Models, to address the heterogeneity in the BitBrains trace caused by the diverse nature of workloads that cloud systems encounter. In this trace, resource usage attributes were targeted for clustering. The results showed better clustering quality for the Gaussian Mixture Model, while less running time was required for the K-means method. Similarly, Mosoti et al. [71] also exploited the K-means method to cluster Grid Workload Archive datasets as part of the virtual machine allocation algorithm called Minimum Interference First Fit (MFF). Both of these works focused on resource usage patterns and left out human factors in their clustering attempts, which can play a significant role in resource management, as discussed in Section 1 (see page 1). Additionally, Ismaeel et al. [72] proposed an approach to predict the required virtual machines using Fuzzy c-means and K-means methods. They concluded that the Fuzzy c-means algorithm can achieve better results compared to K-means.

The majority of the above studies applied cloud trace clustering to improve resource provisioning and utilisation. This is reasonable, as the human aspect was not required for most of these studies. However, human-centred applications, such as steering users towards energy-aware usage, would require the analysis and extraction of humans's behavioural patterns. Thus, it is vital to target the human aspect of workload traces with clustering. This necessitates a comprehensive investigation to test the applicability of clustering for such an implementation, which was not sufficiently discussed in these works.

In addition, the cloud traces in these studies can consist of several attributes, as illustrated in Section 2.1.1, see page 6. These attributes may not all be beneficial for clustering. To deal with the multi-dimensionality in clustering, researchers have used feature (attribute) reduction and selection methods. The authors in [73] presented a novel three-way clustering approach for feature reduction. It achieves this by dividing the targeted data into three regional groups for clustering. The method involves random sampling and feature extraction to introduce randomness and diversity, which makes the algorithm more robust. Such methods reconstruct the original data, which might not be beneficial for cases of behaviour extraction. Similarly, Kang et al. [74] suggested solving the problem of large-scale data representation by designing a multi-domain dimensionality reduction mechanism, the Autoencoder. It is based on the assumption that the process of data clustering in high dimensions should be based on sparsity by simplifying complex data in both spatial and frequency domains. This approach also reconstructs the data features using novel sparse reconstruction to provide further details about data analysis and processing.

In [75], researchers proposed dimensionality reduction for the K-means method through one approach of attribute selection and two extraction methods. This work focused on the theoretical implementation of these approaches without adopting automated techniques. In addition, [76] proposed a mechanism for extracting multidimensional clusters from health databases. This study implemented the dimensionality reduction technique of PCA. It converts the original features into a new set of features called PCA components by extracting the most variation from the original data [77]. In the context of clustering cloud traces, this mechanism leads to generating a new attribute with mixed characteristics of the original features, which may affect the quality of clustering.

Furthermore, Daraghmeh et al. [78] proposed an ensemble clustering approach aimed at revealing complex and hidden patterns in cloud workloads. This is achieved through a combination of different pre-processing techniques and clustering methods. Their approach also utilized PCA to address dimensionality in cloud workloads. In addition, the studies in [15] and [79] developed new subset feature selection and reduction methods. These methods exploited clustering algorithms (such as K-means) for attribute selection and correlation measures for dimension reduction. In this context, feature selection methods may not be beneficial since they select one dimension at a time instead of identifying the best combination of dimensions. Conversely, dimensionality reduction methods (such as PCA) combine features into new ones, which may lead to a loss of their original characteristics.

On the other hand, in [80], researchers used a feature selection method to exclude attributes that are not useful for analysing and predicting resource usage from Google workload traces, such as TIME, COMMAND, Timestamp, and load factor. Similarly, the Recursive Feature Elimination (RFE) method was employed in [81] to improve the accuracy of a proposed prediction model for job failures. Furthermore, the authors in [82] compared several feature selection algorithms, including Improved Reduct and Genetic Algorithms, applied to different learning and heuristic methods for analysing application runtime. These algorithms were used to identify attributes associated with past similar jobs. Bhagtya et al. [83] also utilised the Improved

Reduct method to reduce the number of input workloads from multiple virtual machines, aiming to increase the accuracy of predicting CPU, memory, and disk utilisation and to prevent user migration.

As demonstrated in the above studies, feature selection and dimensionality reduction mechanisms have already been used in various analysis and forecasting applications for cloud computing. The selection process in these applications was either generic or resulted in new features that may not be beneficial for extraction, as mentioned in Section 2.1.2. Furthermore, these studies mainly target the resource aspect of the traces, where the data characteristics are less complex and the necessary selection parameters are known. Hence, the challenge of dealing with complex human patterns in the absence of these parameters was not encountered. In other words, using such general methods of feature selection requires further input parameters (e.g., the number of expected clusters) [18–20]. These parameters are not usually available in cloud workload traces (i.e., they do not disclose the number of users whose utilisation patterns the traces represent).

It is worth noting that the above investigations were conducted using only simple clustering methods (such as K-means). Thus, a wider range of clustering methods was not thoroughly investigated. Consequently, they overlooked one of the main factors affecting clustering quality, which is method selection. Few techniques have been developed to detect in advance the best method among a given set for clustering a particular trace. Most suggested tools are either generic or non-automated. The researchers in [84] presented a selection of clustering methods by evaluating and discussing several modern and traditional approaches. The paper reviews clustering and examines these methods, considering basic factors such as distance or similarity metrics and evaluation criteria. In [85], the authors used the Jaccard coefficient index to compare clustering methods. However, only three clustering methods were evaluated, while others were not investigated. Additionally, [86] compared five clustering methods with 10 validation indices. Nevertheless, the selection of clustering methods was conducted intuitively, with no automated, dataset-oriented technique proposed.

In an attempt to provide a semi-automated approach, researchers in [87] presented an analytical framework to guide the choice of the most suitable clustering algorithm tailored to specific spectroscopy data. It adopts a comprehensive strategy, taking into account the user's requirements, the distinct features of the data, and the potential attributes of the clusters that may emerge. However, this framework still requires human analysis and decision-making. On the other hand, Barak and Mokfi [14] presented an MCDM (Multi-Criteria Decision Making) group model to select and rank clustering methods. In this study, six clustering methods were evaluated using five indices. The authors in [88] also proposed a novel MCDM-based approach named DMSECA (Decision-Making Support for Evaluation of Clustering Algorithms), designed to evaluate clustering algorithms by integrating expert opinions. Similarly, [89] exploited the use of MCDM to examine the effectiveness of different clustering algorithms in the context of financial risk analysis. Nevertheless, these processes and results are descriptive, limiting their suitability for automation, and their outcomes can be difficult to explain and interpret. From the above, we deduce that current techniques are inadequate for identifying high-quality clustering methods for cloud datasets.

### 2.2.2 Exploratory Evaluation of Clustering in Cloud Computing

As mentioned previously, targeting detailed human records, such as the approach proposed by Kecskemeti et al. [4], requires analytical tools capable of extracting and predicting user patterns from cloud traces in a detailed manner. Our discussion above demonstrated that many studies have used clustering to extract vital information from these traces. However, the use of clustering to reveal such detailed characteristics of user records has not been thoroughly investigated. In addition, these studies neglected the most important factors that affect clustering quality: attribute and method selection.

Therefore, in this section, we aim to explore the past literature from two points of view. First, we examine the ability of clustering to extract detailed patterns from workload traces. This also includes testing the use of several clustering methods to determine which are most suitable for extraction. Second, we extend this investigation by providing a comprehensive analysis of the improvement in clustering quality when better attributes and methods are employed. This is carried out by comparing the changes in clustering quality while evaluating relevant methods and cloud attributes. In these investigations, we use the precision metric to validate the clustering results, as it is more appropriate for extraction evaluation, as discussed in Section 2.1.2. These investigations are detailed in the following subsections:

#### Clustering to Reveal Users' Labels

In this investigation, we assessed the ability of clustering to extract patterns of users' consumption records from cloud traces. We performed this by evaluating whether these patterns grouped in a way that each group matched their supposed users' labels. To ensure evaluation efficiency, it is vital to perform an *attribute analysis* to select attributes that are matched and correlated between supervised and unsupervised traces. As a result, if an attribute shows promising accuracy in a supervised trace to identify users, then attributes in unsupervised traces, which correlate with the promising attribute, are expected to show similarly good accuracy. In this context, the supervised traces serve as ground truth to validate the effectiveness of clustering for pattern extraction.

In such evaluation, traces with no user information (unsupervised) are hard to judge in terms of which attribute is good for clustering. Therefore, we check the clustering quality for the supervised trace attributes that have a similar meaning to the unsupervised ones. Consequently, this provides insight into what can be expected for unsupervised traces. This is crucial for the analysis process when only unsupervised traces are available.

Upon completing the analysis process in Section 2.1.2, we continue with the assessment with the following evaluation process: During the initial phase of the evaluation process, we ensure that the clustering algorithms disregard attributes that directly identify user labels from the supervised traces. We remove the user ID attribute from the trace and count the unique user IDs in it. Then, when we apply the clustering methods, we indicate to them that the expected number of clusters is the same as the number of unique user IDs. Finally, we evaluate clustering extraction by comparing whether the clustered patterns are grouped into clusters similar to their disregarded user IDs. The closer the clustering grouping of these attribute patterns is to their original user IDs, the better the extraction. We measure this closeness using the precision metric. It is important to mention that this process will be used in all the extraction evaluations in this thesis.

The above steps involve evaluating several clustering methods to see which ones provide the highest clustering precision for the same correlating attribute. These steps are further elaborated in the following.

## Implementation

a) Attributes analysis: This step is carried out by finding the matching attributes between supervised and unsupervised traces based on their descriptions. We relied on the attribute descriptions provided in Table 2.1 and Table 2.2. This is followed by measuring the correlation between these attributes using $R^2$. Subsequently, nine attributes can be matched between the supervised and unsupervised formats. These relationships are depicted in Figure 2.3, and the reasoning behind these links is detailed as follows:

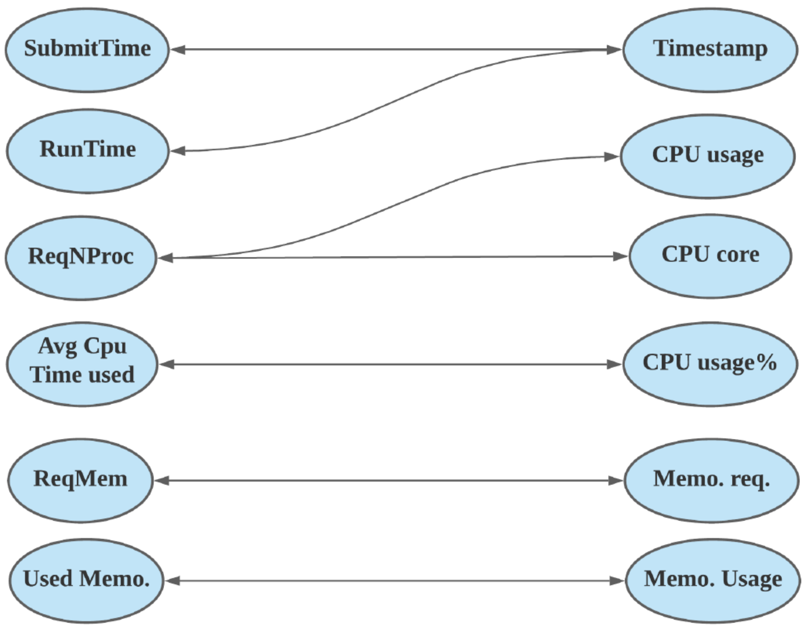


Figure 2.3: Matched attributes between supervised and unsupervised traces.

- Submit time/Run time can potentially be matched and correlated with timestamp, as they show a Pearson correlation value of 0.9. All of these can be considered to characterise the times when the user's jobs are active.
- Requested/Allocated number of processors have potential links with CPU cores/usage, as they depict user requirements for CPU count, with a Pearson correlation value of 0.8.
- Requested/Used memory is matched with memory provisioned /usage (requested), as these differ only in level of detail (the Materna and Bitbrains traces offer this at finer granularity), with a correlation value of 0.4.

b) Evaluation Process: As the *second step*, each matched attribute was clustered individually to test its ability to identify the user label of the workload trace. To this end, five supervised traces from Table 2.4 were used as samples to assess the impact of method selection on clustering quality. These traces are (i) NorduGrid, (ii) SHARCNet, (iii) DAS2, and (iv) HPC2N-2002. These traces were chosen randomly to avoid any bias.

Firstly, we carried out a comparison test between the DAS2 and HPC2N-2002 traces using 10K individual lines of records randomly selected from each trace. We applied several clustering methods that are widely used in related works (i.e., [68] and [73]), namely: Fuzzy c-means, Simple EM, Hierarchical, and K-means. Then, we used these methods with each previously selected correlated attribute on both traces. These methods were selected because each represents a type of clustering category described in Section 2.1.2 (e.g., Partitional and Hierarchical). Accordingly, the attributes Submit Time, Run Time, Used Memory, and Requested Number of Allocated Processors (ReqNProc) were used individually with each clustering method. The attribute Requested Memory was not considered when judging the suitability of the clustering methods because many traces contained too many records with this attribute unspecified (with a value of -1). We narrowed down the clustering methods to those that identify clusters most related to the user IDs. Then, the method that gives the best-performing clustering result was used to cluster the other traces' attributes of GWA-T-3 NorduGrid, GWA-T-10 SHARCNet, and METACENTRUM-2013 to demonstrate their suitability for extraction.

### Experimental Results

The results of these samples are shown in Figures 2.4 and 2.5. The figures demonstrate that the K-means clustering method gives the highest clustering precision compared to Fuzzy c-means, Simple EM, and Hierarchical methods. This indicates that K-means can better identify that a particular trace line belongs to a specific user. However, such a result does not necessarily mean that the K-means method is superior. Nevertheless, the rest of the evaluation was carried out using this method for clustering other traces, as it was observed to be the easiest to use.

In the next phase of the investigation, 5K randomly selected trace lines were evaluated using the K-means method to represent a sample demonstrating the impact of selecting different attributes on clustering quality. As before, the number of expected clusters was predefined based on the unique user IDs found in the trace line sample. Unfortunately, not all matched attributes in the traces contained usable information for clustering as many were filled with -1 values in large portions, as discussed above. The attributes used for the evaluation are indicated with check marks (✓) in Table 2.13.

**Findings** The results above indicate that the attributes with the highest clustering precision are those representing what has been requested from the infrastructure, such as *Requested Memory* and *Used Memory.* Thus, these attributes offer the greatest potential to reveal users' patterns and

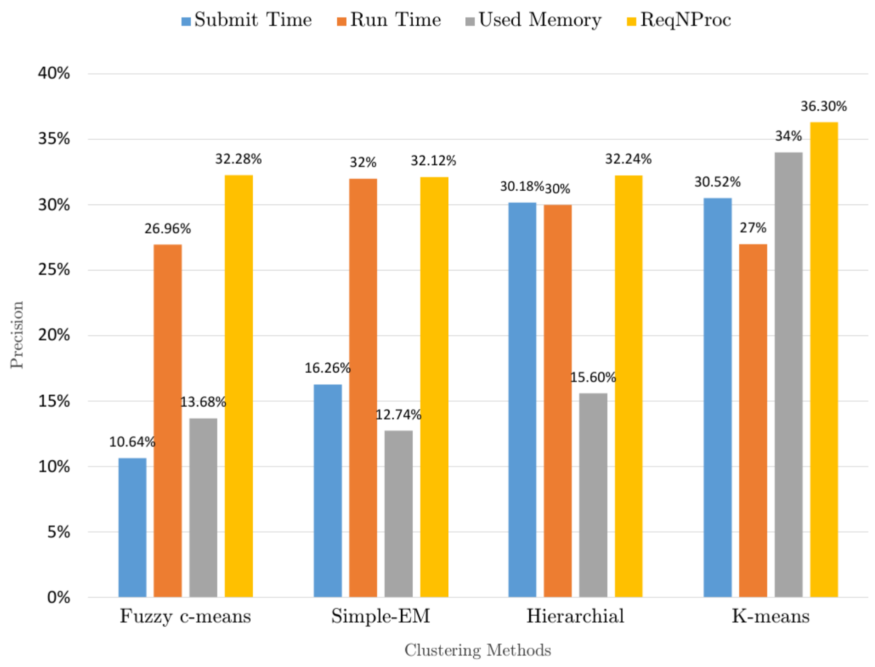


Figure 2.4: Sample showing the impact of method selection on clustering quality in the DAS2 trace

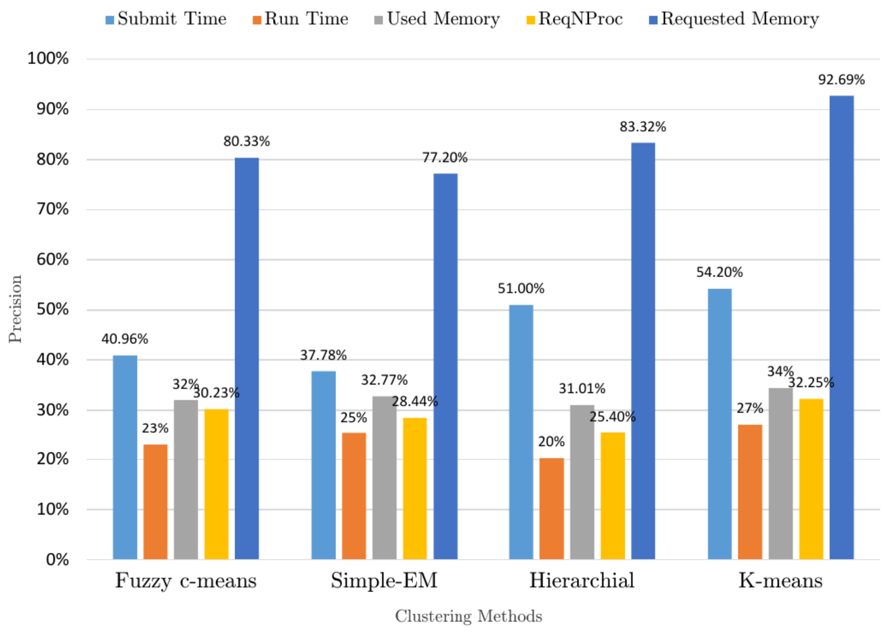


Figure 2.5: Sample showing the impact of method selection on clustering quality in the HPC2N-2002 trace

Table 2.13: Considered correlated attributes for user identification

| Trace | Submit Time | Run Time | Used Mem. | Req. Mem. | Req. N Proc |
|---|---|---|---|---|---|
| DAS2 | ✓ | ✓ | ✓ | × | ✓ |
| Sharcnet | ✓ | ✓ | ✓ | ✓ | × |
| Nordugrid | ✓ | ✓ | ✓ | × | × |
| HPC2N-2002 | ✓ | ✓ | ✓ | ✓ | ✓ |
| META-2013 | ✓ | ✓ | × | ✓ | ✓ |

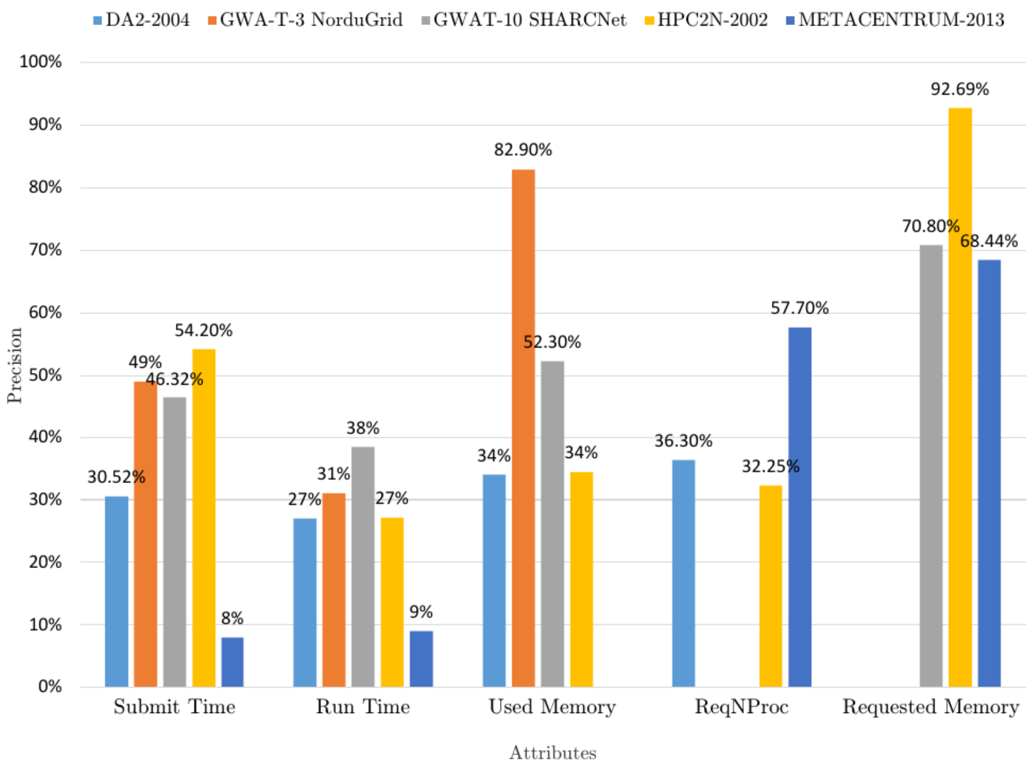


Figure 2.6: Sample showing the impact of attribute selection on clustering quality

enable identification. The results also show that the attribute *Submit Time* provides lower clustering precision. This is due to the incremental nature of its value distribution across the dataset, which reduces its ability to reveal detailed patterns in the trace. Accordingly, it can be inferred that user patterns in workload traces can be extracted using clustering methods with high precision of around 92% in some cases. However, further investigation is required to determine the factors affecting clustering quality.

**Impact of Attribute and Method Selection on Clustering Quality**

This subsection extends the previous investigation. It aims to examine the potential enhancement in clustering quality by selecting a more effective clustering method or attributes. This includes specifying the criteria for selecting applicable methods and the cloud traces used in the test.

In the first experiment, we tested the improvement in clustering quality by incorporating all the applicable attributes from the cloud traces provided in Table 2.4. We used the K-means clustering method, as it showed high compatibility with cloud traces, as demonstrated in the previous investigation. In the second experiment, we repeated the assessment in Subsection 2.1.2, this time employing all relevant clustering methods. To do so, we included additional clustering methods in the comparison tests. We selected these methods based on the following criteria:

- **Set the number of clusters**: The method should allow for the predetermination of the number of clusters. This ensures that all methods cluster into an identical number of groups, facilitating accurate comparison.
- **Results comparability**: The methods should yield clustering results in the same format (i.e., numerical labels in our case). This ensures comparability across all selected methods without the need to alter their formats.
- **Pre-defined parameters**: The pre-defined parameters required by these methods should be as similar as possible, to avoid any biased conclusions.

For the clustering method comparisons, we have selected the methods listed in Table 2.14, all of which meet the specified criteria above. This represents a more evaluation-focused selection process compared to the method selection used in the previous investigation on page 38.

In both experiments, we set the predefined number of clusters to 49+1: 49 clusters representing individual users and their trace lines, and one additional cluster for all other users' trace lines. As observed in Section 2.1.2, it is rarely possible to identify more than 50 users with high accuracy from these traces (see page 2.1.2). Similar to the evaluation process in Section 2.1.2, we used clustering to extract user patterns from supervised cloud traces, after removing the attributes that explicitly indicate user identities (i.e., labels). This was performed for all applicable attributes across all traces listed in Table 2.4. Based on clustering precision, we assessed the improvements achieved

Table 2.14: Clustering methods selected for use in this dissertation

| **Name** | **Abbreviation** |
|---|---|
| K-means | - |
| X-means | - |
| Farthest First | F. First |
| Fuzzy C-means | - |
| Self Organised Map | SOM |
| Learning Vector Quantization | LVQ |
| Cascade K-means | - |

by the applied attributes and methods. These experiments are described in more detail below:

### Impact of Attributes Selection

To calculate the improvement from the first experiment, we identified the highest precision recorded for single-attribute clustering for each trace in Table 2.4. We then compared these results with the highest precision achieved by multi-attribute clustering for the corresponding traces. This comparison was conducted to ascertain the maximum impact of applying all possible selections of attributes. The explanation below demonstrates the procedure of this experiment and presents its findings.

**Dual-Attribute Clustering** We started this experiment with the simplest dimensionality: dual-attribute clustering. In this experiment, we clustered all possible attribute pairings for all the traces in Table 2.4. From these combinations, we identified the pairs that achieved the highest precision for each trace. By calculating the difference between the highest precision of single-attribute and dual-attribute clustering results, we measured the highest possible improvement. The results demonstrate a noticeable increase in dual-attribute clustering precision for some traces, while others exhibit a decrease or no improvement; see Figure 2.7. It was observed that the greatest improvement was around 15%, recorded for the pair (Number of Allocated Processors and Run Time) in the CIEMAT Euler trace. This increased from 40% precision recorded for single-attribute clustering of the same trace. Conversely, the dual-attribute clustering of (Used Memory and Wait Time) in PIK IPLEX showed the largest drop, of about 9%.

We analysed these results by comparing the statistical measure of skewness for each attribute pair with their clustering ground truth (user IDs) in

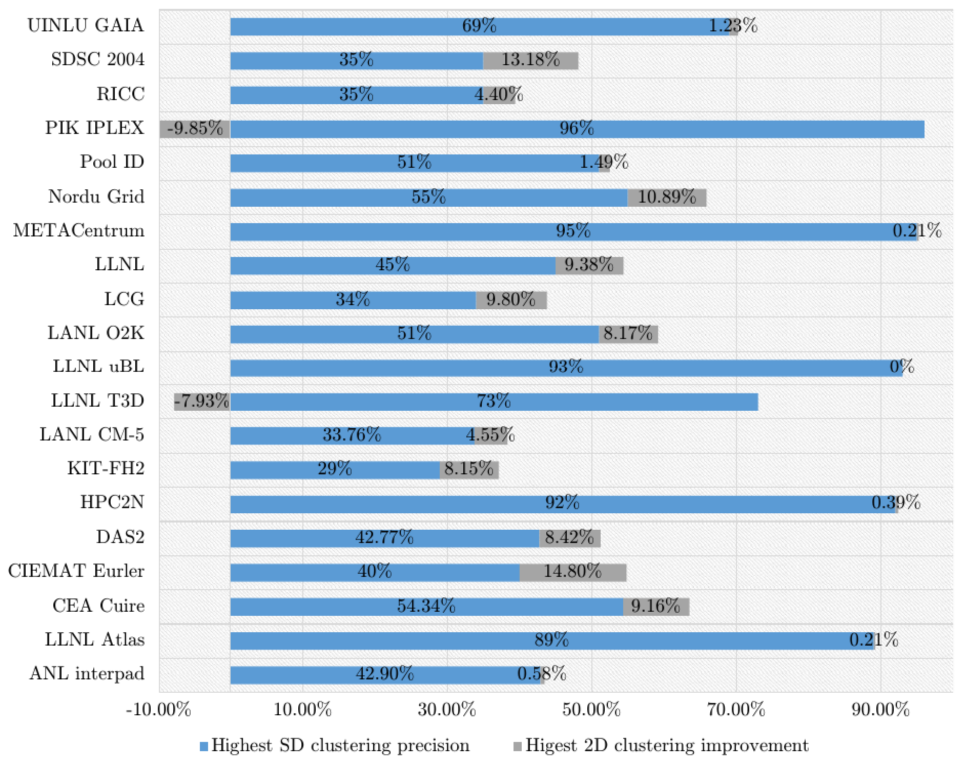


Figure 2.7: Highest improvement from single-attribute (SD) to dual-attribute (2D) clustering across all applicable traces

the corresponding trace. As previously mentioned, we used clustering to extract user IDs from these traces. For instance, in the CIEMAT Euler dataset, the skewness for Number of Allocated Processors was 21 and for Run Time was 19, compared to a skewness of 20 for the target User ID. Our observations indicate that attributes with skewness values closer to their target tend to show more improvement. Conversely, the greater the difference, the smaller the improvement. This pattern holds true for the pair Used Memory and Wait Time in the PIK IPLEX 2009 dataset. This suggests that the proximity of skewness values between attributes and their clustering ground truth can lead to better quality. It is important to note that this analysis cannot be conducted for general-purpose clustering aimed at extracting hidden information, where no ground truth is available. Hence, a detection technique may be necessary.

**Multi-Attribute Clustering** We further evaluated the improvement in clustering quality by extending from dual-attribute to triple-attribute clus-

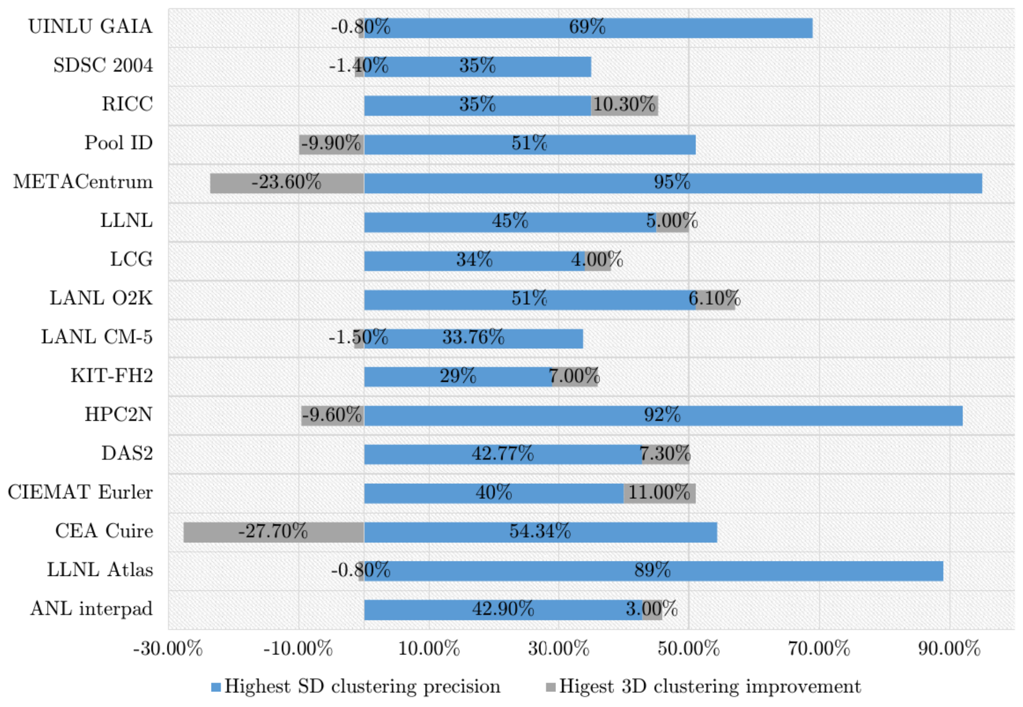


Figure 2.8: Highest improvement from single-attribute (1D) to triple-attribute (3D) clustering across all applicable traces

tering. For this experiment, we selected the attributes with skewness values closest to those of the targeted user IDs for triple-attribute clustering. We disregarded traces that did not have enough applicable dimensions for triple-attribute clustering, as shown in Table 2.4. Figure 2.8 shows the triple-attribute combinations that recorded the highest clustering improvements for each trace.

The combination of attributes (Submit Time, Wait Time, and Average CPU Time Used) in CIEMAT Euler showed the highest improvement of around 11 percentage points compared to clustering them individually; see Figure 2.8. However, this is lower than the highest dual-attribute clustering precision for this trace, as shown in Figure 2.7. In contrast, the combination of attributes (Requested Time, Wait Time, and Run Time) for CEA Cuire exhibited the biggest drop, recording 27% precision, down from 54% for their dual clustering. The precision was 63% for the dual-attribute clustering of (Wait Time and Run Time) without Requested Time. This highlights the importance of selecting the right combination of attributes for clustering.

**Findings** After conducting a thorough investigation, it was observed that the highest improvements can be achieved through dual-attribute clustering. However, as the number of attributes increases—for example, in triple-attribute clustering—the effectiveness of clustering decreases. This trend was further tested with samples of four- and five-attribute combinations. The findings indicated that as more attributes are added, the likelihood of improving clustering decreases. This suggests that arbitrarily increasing the number of attributes may lead to overfitting the clustering model, thereby reducing its quality. Despite this, the investigation revealed that a significant improvement in clustering performance—around 10% to 15%—can be achieved by selecting the right combination of attributes. However, attaining this improvement requires a more meticulous approach to detecting and utilising these optimal combinations.

### Impact of Clustering Method Selection

For the fourth experiment, we repeated the same assessment process described in Section 2.2.2 (see page 36). In this experiment, the evaluation was conducted by clustering the attributes (Submit Time, Wait Time, Run Time, Number of Allocated Processors, and Average CPU Time Used) using the selected methods listed in Table 2.14. We targeted all attributes available across these traces. Below, we provide more detail on the implementation and findings.

**Experimental Implementation** We randomly selected two traces (Unilu Gaia and DAS 2003) as samples. For Unilu Gaia, the lowest precision was below 20% when clustering Run Time using Cascade K-means, as shown in Figure 2.9. This increased to around 90% by clustering the same attribute with either the F. First or LVQ methods. Similarly, the precision increased from 19% to 89% by clustering the Number of Allocated Processors with these two methods instead of Cascade K-means. These results demonstrate an improvement of up to 70% when using a more effective method.

The results for DAS 2003 show contrasting performance between F. First and LVQ when clustering the attribute Run Time. As shown in Figure 2.10, the precision was below 38% for these methods, while it increased to 55% using the X-means method. Additionally, the precision increased from 41% when clustering Average CPU Time Used with the LVQ method to 68% when using a more effective method, K-means (see Figure 2.10). This illustrates that a clustering method may be effective for a particular attribute in one trace yet ineffective for the same attribute in another.

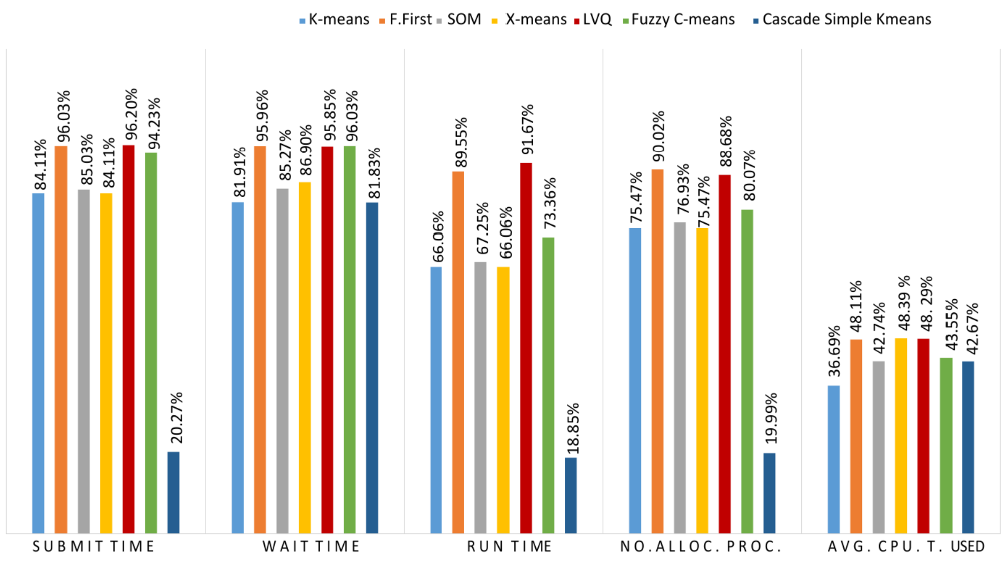


Figure 2.9: Impact of method selection on clustering precision based on input attributes in the Unilu Gaia trace

**Findings** It can be inferred from the results above that selecting more effective methods can lead to better clustering quality. This improvement can be significant—up to 70 percentage points. However, such selection cannot be done arbitrarily. Therefore, it is vital to have a technique that can detect in advance which clustering method is more effective for a particular trace.

### 2.2.3 Discussion

The findings of this section showed that relatively high precision can be achieved in extracting detailed patterns from cloud traces using clustering methods. The results also showed that attributes related to user demands provide the highest clustering quality compared to other attributes. The later investigation also demonstrated that extraction can be improved significantly when choosing a suitable clustering method and attributes for the targeted trace. This makes it crucial to provide an automated technique to accomplish these tasks in advance.

## 2.3 Review of Forecasting in Cloud Computing

In the area of cloud computing, researchers have developed various forecasting models for different purposes. Most of these models specifically aim to

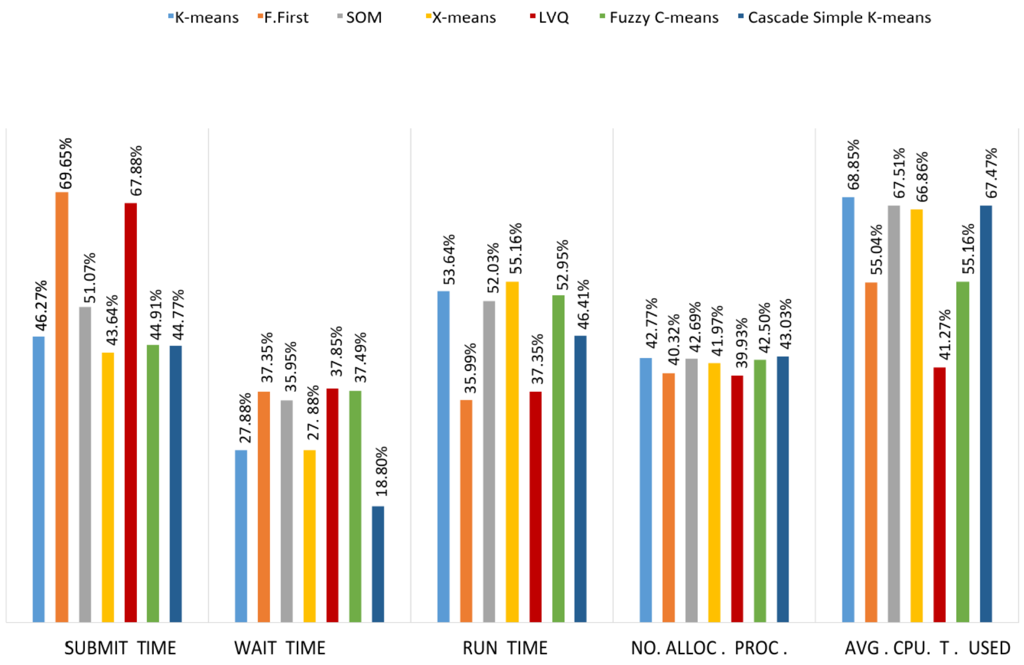


Figure 2.10: Impact of method selection on clustering precision based on input attributes of the DAS 2003 trace

address the challenges of dynamic resource management and scaling.

In [22], Lu et al. proposed a model called RVLBPNN to forecast workload trends based on historical data, combined with the workloads' level of latency sensitivity. Later, [90] presented an improved version of RVLBPNN by incorporating the K-means clustering method. This new version predicts future workload trends based on the history of response time characteristics for these workloads.

Maiyza et al. [91] also aimed to predict workload values and future trends by presenting VTGAN, a non-linear prediction model. In [92], Arbat et al. proposed a time-series forecasting model designed to predict changes in cloud workloads with high accuracy and low inference overhead. The model, called WGAN-gp Transformer, is inspired by the Transformer network and improved Wasserstein-GANs. It aims to address the challenges posed by the dynamic nature of cloud workloads.

Kumar et al. [93] developed an LSTM/RNN-based model to enhance resource management and optimise performance by accurately predicting future workloads, which is crucial for efficient operation in cloud environments. The model predicts workload values based on previous samples. The authors also presented a similar forecasting approach in [94], embedding a self-directed learning process to predict future demand from cloud servers.

Likewise, in [21], Zhang et al. developed a MAG-D model based on a GRU neural network. This model predicts memory and CPU usage of each cloud resource based on data centre traces. On the other hand, to forecast user behaviour trends in large-scale cloud environments, Panneerselvam et al. [95] implemented the InOt-RePCoN model. This model aims to predict the number of jobs and submission times for users. Similarly, in [96], Nehra and Kesswani presented an LSTM-based forecasting model to predict workloads in a cloud computing environment, aiming to reduce service level agreement violations.

On the other hand, Qiu et al. [97] introduced an advanced large language model (LLM) tailored for energy forecasting tasks (e.g., electricity load, solar, and wind power). In this study, the authors aimed to address the challenge of what is called *hallucination*. In this context, hallucination refers to cases where forecasting models produce outputs that seem plausible but are not grounded in actual user behaviour. This occurs when applying unstructured and ambiguous data, similar to the data presented in Table 2.3, which we disregarded in our dissertation to avoid such a phenomenon.

It can be concluded from the above that a similar forecasting approach is followed by the models in these studies. This approach targets and macro-predicts the overall values and trends of workloads. Such a methodology is designed to treat users' patterns in the traces as a whole. Unfortunately, the traces in their raw form do not reflect any meaningful patterns for prediction. Thus, the current models lack an efficient mechanism to uncover and capture the diversity and variability of users' consumption at a detailed level.

## 2.4 Summary

This chapter covered the background of extraction and forecasting tools for analysing cloud users' behaviour, including the investigation environment for each of them. It also gave an overview of applying these tools to related cloud computing work, highlighting their gaps and limitations. This was followed by a comprehensive literature review through two investigations. First, we tested the extraction ability of clustering methods by checking how they can group users' patterns in a way that is similar to their labelling, without depending on the attributes that directly reveal these labels. Then, we extended the first investigation by checking the improvement in clustering quality when selecting a better method and attributes. The findings presented in this chapter have demonstrated the gaps we aim to close in this thesis.

# Chapter 3

# Research Methodology

## 3.1 Introduction

To provide efficient analysis tools, we investigated the effectiveness of clustering methods for data extraction and reviewed the approaches used in related works' forecasting models, as detailed in Section 2.1. The findings have shown that high accuracy in extraction can be obtained by selecting suitable clustering attributes and methods. For attribute selection, the commonly exploited tools in the literature (i.e., dimensionality reduction and feature selection) either require inputs that are not always available in cloud traces [17–19] or tend to produce new attributes from the originals. For clustering method selection, most of the proposed tools are descriptive with no fully automated process. In addition, our review also demonstrated that the related works' models were structured to macro-predict users' patterns as an overall trend, which is not suitable for detailed forecasting.

To address the above gaps, we introduced the following three tools to meet the goals in Figure 1.1: non-supervisory attribute selection, clustering multi-attribute combination and method pre-detection, and the micro-prediction approach. First, we presented an unsupervised method of attribute selection specialised for cloud workload traces. This method aims at overcoming the limitations associated with parameters that are not readily available in these traces. In the new method, Sequential method of clustering Quality (SeQual), we exploited the use of the SC metric to measure the quality of clustering each attribute with a range of predefined numbers of k inputs. As a result, the clustering result for each k of the corresponding attribute forms a scale. Based on this scale, the attributes are ranked. Accordingly, SeQual addresses clustering dimensionality by focusing on selecting the best single attribute for a given clustering method.

Table 3.1: Performance comparison of the validation metrics

| **Quality metric** | **Accuracy** | **Run Time (seconds)** |
|---|---|---|
| Davies–Bouldin | 87% | 20 |
| Silhouette Coefficient | 79% | 195 |
| Calinski-Harabasz | 50% | 20 |

Next, we proposed the second tool, Effectiveness detection of clustering quality (EFection), to pre-detect attribute combinations and methods for clustering. Our technique performs this in an unsupervised manner, without the need to analyse the full traces. EFection achieves this by analysing samples of these traces using both $R^2$ and other internal validation metrics while applying various clustering methods to these samples.

Third, we developed an approach that can micro-predict detailed consumption patterns of cloud traces that are not usable, in their default form, for such forecasting due to their characteristic of sudden changes. Our approach conducts this for both uni-attribute and multi-attribute forecasting scenarios. To do so, it employs the above extraction tools to capture these patterns in clusters.

Then, in the second phase, these clusters are uniformed with time alignment and linear interpolation to be used for the training process. This produces a trained model for each cluster of the corresponding trace. These models are then used in the forecasting. Accordingly, we present this approach as Micro-forecasting approach for cloud user consumption pattern based on RNN (MICRAST). The sections below illustrate in detail the description of our tools.

## 3.2 Extraction tools

This section provides a detailed description of our tools, which aim to ensure more efficient extraction through clustering and achieve the goals depicted in Figure 1.1. These tools are the non-supervisory method for attribute selection (SeQual) and a technique for clustering attribute combinations and method pre-detection (EFection).

### Clustering Internal Validation Metric

It is vital to select an appropriate validation metric for clustering when developing pre-detection and selection techniques. We aim to reveal common patterns of cloud user behaviours using unsupervised clustering. Therefore, it is necessary to select an internal validation metric that offers an efficient quality measure without relying on external criteria. To achieve this, we began the development of extraction tools with a preliminary comparison between three well-known internal validation metrics (DB, SC, and CH).

We evaluated the performance of each metric by calculating and comparing their accuracy and run time to assess clustering quality. We consider run time important since the selection process in EFection requires multiple iterations of different quality measures for each recommended combination of attributes and clustering methods. This makes run time a crucial factor when selecting an internal validation metric for EFection, whereas in SeQual, this was less critical because the validation is performed on a sequence of single attributes.

The clustering here aims to reveal users' labels in cloud traces by grouping granular consumption patterns into clusters after disregarding the attributes that directly reveal these labels (i.e., user ID). This approach is similar to the investigation experiments in Section 2.1.2. First, we used an external validation metric (i.e., precision) to measure clustering quality. In this context, precision is measured by comparing how closely the clusters of detailed patterns correspond to their labelling in the omitted user IDs. Then, we applied each of the DB, SC, and CH metrics separately to measure the quality of the same clusters internally, without relying on external criteria. Finally, we calculated the accuracy of these metrics by comparing them to the precision for each clustered record, along with their run times. This involved calculating the percentage error using the following equation:

$$\text{Percentage Error} = \left( \frac{|\text{Observed Value} - \text{True Value}|}{|\text{True Value}|} \right) \times 100$$

A percentage error refers to the percentage deviation from the true or agreed-upon value, expressed as the difference between the experimentally obtained measurement and the and the reference value. It indicates whether there is any deviation and how precise a measurement or estimation is [98].

We drew boxplots showing the distribution of percentage errors for each metric to compare their certainty. Based on these results, we selected the internal validation metric with the relatively highest accuracy and lowest run time to ensure usability. Table 3.1 and Figure 3.1 demonstrate the results of this experiment. The accuracy for the DB was 87%, with percentage errors ranging between 3.3% and 11.3%, and a run time of only 20 seconds. The SC showed higher certainty, with percentage errors ranging from 0.5% to 9.1%, an accuracy of 83%, but a relatively longer run time of around 165 seconds. Meanwhile, CH recorded only 50% accuracy with a run time of 20 seconds. The percentage errors for this metric were distributed between

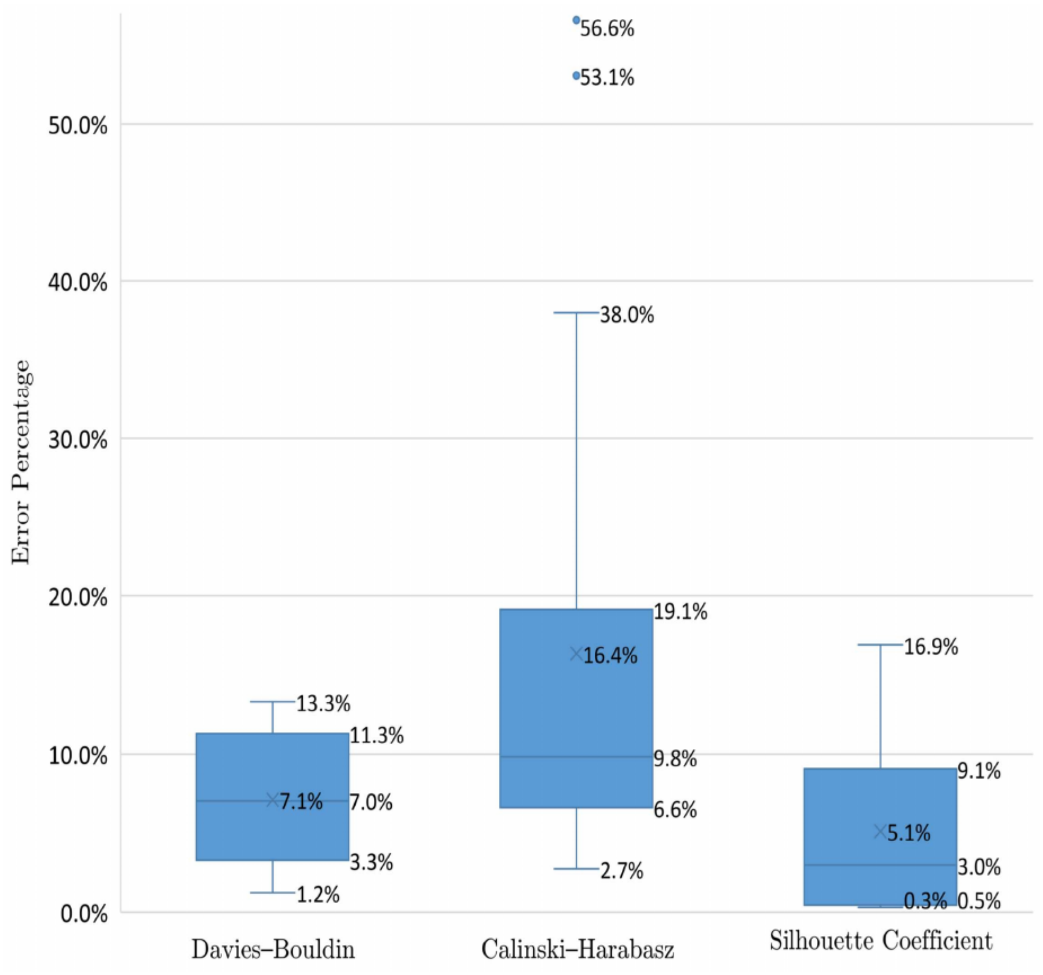


Figure 3.1: Distribution of percentage errors for the internal validation metrics

6.6% and 19.1%, with two outliers at 53.1% and 56.6%. It is evident from the results above that the DB metric offers faster execution with high certainty, making it suitable for techniques that require more intensive computation. In contrast, the SC metric demonstrated more accurate results, albeit with relatively slower execution, making it more appropriate for techniques with simpler computational demands.

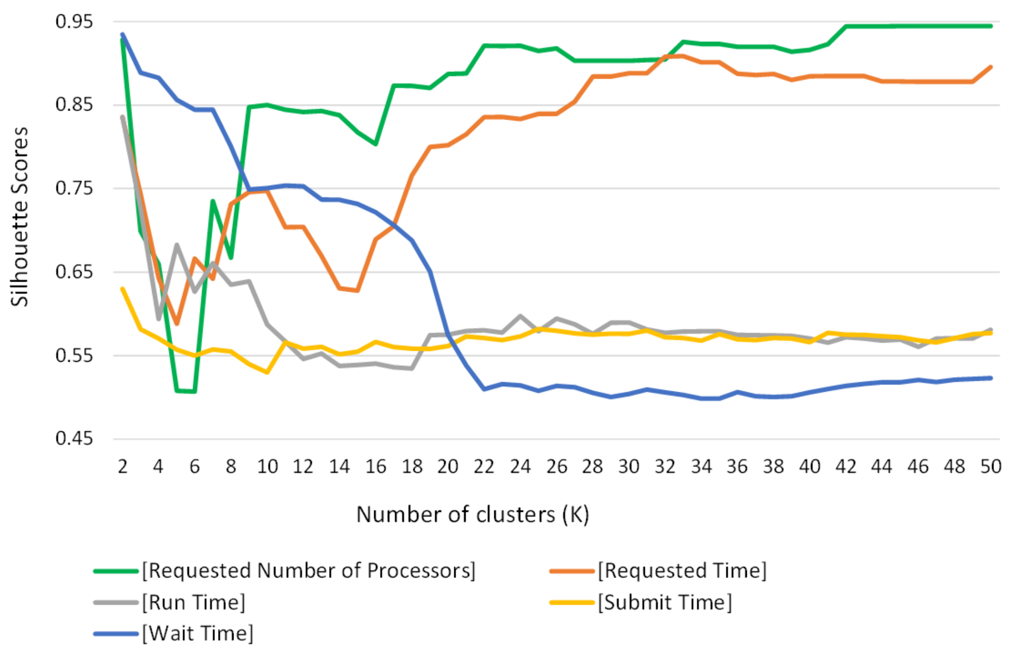


Figure 3.2: Behaviour of SC scores for all applicable attributes in the ANL-Interpad 2009 trace

### 3.2.1 SeQual: Attribute Selection Method for Clustering Cloud Workload Traces

This method selects a clustering attribute for each trace without asking for supervisory inputs or full trace analysis. It conducts this by first clustering samples for each attribute. The clustering process is repeated for each attribute for a predefined number of clusters, which ranges from 2 to 50, or any reasonable range defined by the user. We have used this range, as in the supervised traces where user identification is available, we have observed that it is rarely possible for a trace to exhibit more than 50 uniquely identifiable patterns, as illustrated in Section 2.1.1.

Then, SeQual measures the quality of these clustering results at each point in the above range using an internal validation metric. For this purpose, we selected the silhouette metric, illustrated in Section 2.1.2, since our technique required a simple computation process. As a result, this forms a sequence of SC scores from 2 to 50 for each attribute. By drawing a plot of this sequence, a scale of quality is obtained for each attribute.

Figures 3.2 and 3.3 show a sample of such scale of SC quality for the

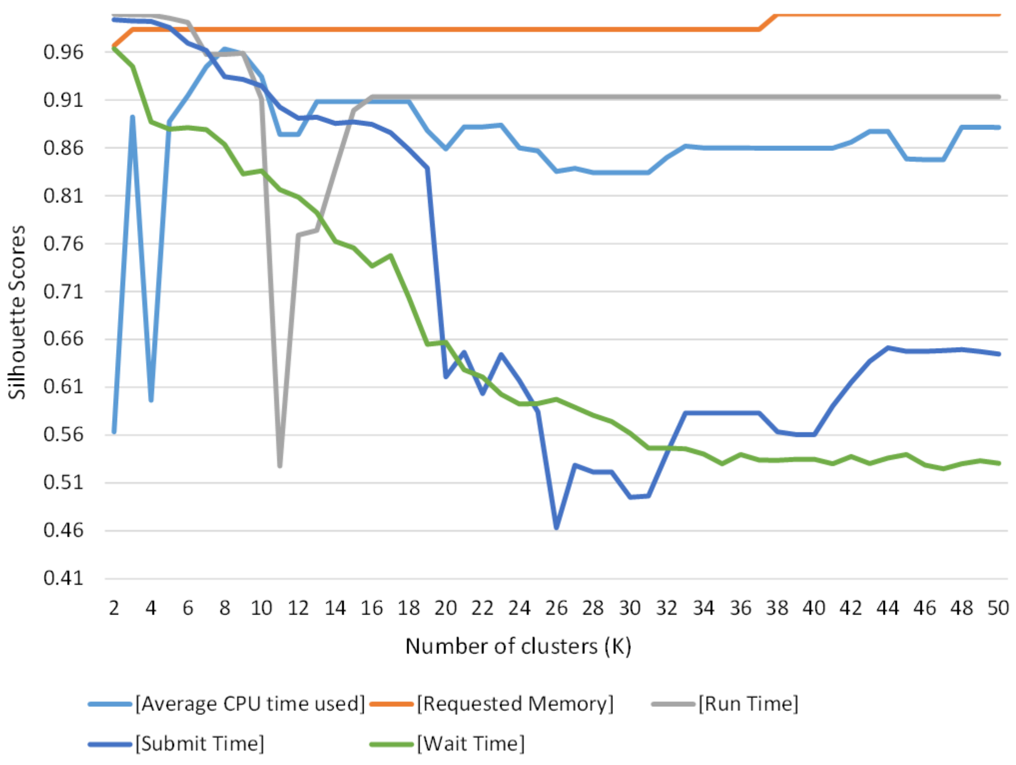


Figure 3.3: Behaviour of SC scores for all applicable attributes in the PIK Iplex trace

most relevant attributes from two sample traces (the ANL-Interpad 2009 and PIK-IPLEX). Based on these charts, the attributes, Requested Number of Processes, Requested Time, Run Time, Submit Time, and Wait Time, each show different scales in terms of silhouette quality.

For instance, in 3.2, the attribute of Requested Number of Processors shows a higher average of silhouette in comparison to the attribute of Submit Time. The figure also demonstrates noticeable peaks and troughs in the entire range of the measurement. In this context, the peaks and troughs represent irregular and sudden changes in clustering quality. Such behaviour can also be seen in the SC scales of other trace attributes. In Figure 3.3, the attributes show slightly different Silhouette behaviour. For instance, the attribute of Run Time shows a quality scale of sharp trough and peak and more sustainable quality compared to the gradually decreasing one of Wait Time.

Based on these observations, we concluded that any attribute showing a high mean of SC with a sharp peak and trough will potentially have a high ability to extract detailed consumption patterns. As a result, the attributes with such behaviour will be ranked higher for extraction, while those with gradually increasing or stable trends will be ranked lower. This was used to devise the proposed SeQual method for attribute ranking and selection.

We will describe the SeQual method in algorithm 3.2.1. In the first step, the user selects a trace $T$ for attribute selection (Step 1). The trace $T$ consists of a set of attributes $C = \{c_1, c_2, \ldots, c_m\}$, and each attribute $c_n$ is processed individually. For each attribute $c_n$, the algorithm performs clustering using the K-means method, considering values for $k$ ranging from 2 to 50 (Step 5). The results of the clustering process are stored in $\boldsymbol{C}_{n,k}$. After clustering, the quality of the clustering is evaluated using the silhouette coefficient $S_{n,k}$, which measures how well-separated the clusters are (Step 6).

**Algorithm 1** The proposed SeQual method

```
Input: $T$, input traces; $C = \{c_1, c_2, \ldots, c_m\}$, attributes from $T$    Output:
Ranked attributes $[C, \mathbf{r}]$ with corresponding ranks.
 1: Start
 2: $n \leftarrow 1$
 3: while $n \leq |C|$ do
 4:     for $k \leftarrow 2$ to 50 do
 5:         $\boldsymbol{C}_{n,k} \leftarrow \text{Cluster}(c_n, k)$
 6:         $S_{n,k} \leftarrow \text{Silhouette}(\boldsymbol{C}_{n,k})$
 7:     end for
 8:     Quality (scale, average): Plot the silhouette score $S_{n,k}$ for $k = 2$
    to 50 for each $c_n$ and compute their average.
 9:     $n \leftarrow n + 1$
10: end while
11: $\mathbf{r} \leftarrow \text{Rank}(C, \text{Quality (scale, average)})$        ▷ Rank attributes based on
    silhouette score analysis (scale, peaks, troughs, and average)
12: Return $[C, \mathbf{r}]$
13: Stop
```

This process of clustering and calculating the silhouette score is repeated for all values of $k$ from 2 to 50 for each attribute. Next, we measure the $Quality(scale, average)$ for each attribute $c_n$ by plotting the silhouette scores $S_{n,k}$ for each value of $k$ and calculating the average of these scores for each attribute (Step 8).

Once the $Qulality$ $(scale, average)$ measured for all the attributes, they are analysed for ranking based on that. The attribute ranked highest if it exhibited $Quality(scale, average)$ with the sharpest peaks and troughs and the highest average of silhouette. A peak and trough are identified by finding a significant drop in the silhouette score over fewer than three clusters (Step 11). This entire process is repeated for each attribute $c_n$ in the input trace $T$. Finally the ranked list of attributes along with their corresponding ranks are returned (Step 12).

As illustrated, SeQual aims at addressing challenges of ranking the best single attribute for clustering without need for supervisory inputs or analysing full trace. However, the dimensionality reduction aspect, where there is a need to detect which combination of attributes is best for the extraction, requires further consideration. Therefore, we propose the detection technique in the next subsection.

### 3.2.2 EFection: Effectiveness Detection Technique for Clustering Cloud Workload Traces

This subsection describes the process leading to the EFection technique. This technique aims at offering the ability to detect the combination of attributes and the method that likely to give the highest clustering quality. It provide such detection without changing the original data, ensuring entire automation process.

**Algorithm 2** The proposed EFection technique

```
Input: $(M, T)$
Output: $k_{\max}, m$
 1: $T_f \leftarrow$ Filtering $T$
 2: $T_s \leftarrow$ uniform of random selection $T_f$
 3: $D \leftarrow \emptyset$
 4: $C_s \leftarrow \{\forall c_i, c_j \in T_s : R_2(c_i, c_j) > 0.5 \wedge c_i \neq c_j\}$
 5: for $\forall m \in M$ do
 6:     if $C_s \neq \emptyset$ then
 7:         $K \neq P(C_s) \setminus \emptyset$
 8:         for $\forall k \in K$ do
 9:             $D \leftarrow D \cup \{(k, \mathrm{DB}(c_{\text{clustering}}(k, m)))\}$
10:         end for
11:     else
12:         for $\forall c_i \in D$ do
13:             $D \leftarrow D \cup \{(c_i, \mathrm{DB}(c_{\text{clustering}}(c_i, m)))\}$
14:         end for
15:     end if
16: end for
17: Return: $(k_{\max}, m : d \in D : \mathrm{DB}(c_{\text{clustering}}(k_{\max}, m)) = m_{\max})$
```

**EFection Description**

The results from the literature review in Figures 2.9 and 2.8 highlight the importance of selecting an effective method and attributes to ensure high clustering quality. Therefore, we proposed a technique to pre-detect such selection. Algorithm 2 provides a detailed description of our technique, EFection. In this algorithm, the user initially inputs the clustering methods and cloud trace into $M$ and $T$, respectively. We expect the user to select these methods and traces upon meeting the characteristics in Tables 2.4, 2.5, and 2.14. No further parameters are required, and our technique does not ask for a clustering validation criterion. By default, EFection considers all the attributes in that trace and all the selected clustering methods, but the user can limit the choice. Our technique's algorithm proceeds in three main phases:

- *Pre-processing phase*: The algorithm starts with pre-processing from steps 1 to 3. It begins by filtering the provided trace $T$ to $T_f$ in step 1. The filtering process is conducted by disregarding attributes with a distribution of constant values above 80%, as they are deemed unsuitable for clustering according to the discussion in Table 2.4. Then, our algorithm randomly selects the sample portion $T_s$ from the filtered trace $T$ in step 2. Each attribute in $T_s$ is denoted by $c$. Each $c$ is a vector of variables $(v_1, v_2, ..., v_n)$, essentially a column in the filtered trace, where $n$ is the size of $c$. Sampling each attribute randomly ensures that the next phases will be less biased towards any particular part of the workload trace.

- *Analysis phase*: This phase extends from steps 4 to 16 to analyse the filtered attributes and the provided methods for potential detection. In this phase, the algorithm measures the $\text{R}^2$ for all combinations of pair attributes ($c_i, c_j$ in $T_s$ 4. If the $\text{R}^2$ between any pair of c exceeds 0.5, the corresponding pair is stored in $C_s$, which is a subset of $T_s$ 4. As mentioned previously in the description of methodology, we used $\text{R}^2$ for filtering the combination attributes since it can determine a strong linear correlation between these attributes. Such a strong correlation can lead to an improvement in clustering quality when it reaches above 0.5, based on our observations. After identifying the combination of attributes that exhibit strong correlations, the algorithm performs an outer loop of the size of $M$ 5, for each clustering method m in M. Within this loop, for each clustering method, if $C_s$ is non-empty, the algorithm finds the power set of $C_s$ (encompassing all potential attributes of $c_i$ in $C_s$). These attributes are stored as subsets in the $K$ set, excluding the empty set. This step gathers all possible combinations of attributes that are recommended. The absence of the empty set implies that there are no combinations selected, suggesting that clustering the attributes individually is preferable.

If there are recommended combinations (i.e., $C_s$ is not empty), an inner loop 8 starts to measure the DB score for clustering each dimension of combinations in $K$ with the method $m$ 9. Each score, along with its corresponding dimension and clustering method, is catalogued in the set $D$. Otherwise, if $C_s$ is empty, the inner loop of line 12 measures the DB score for clustering each attribute $c_i$ in $T_s$ individually. These scores, along with their respective attributes and clustering methods, are then stored in the set $D$ 13. In this phase, both $\mathrm{R}^2$ and the DB metric work together to analyse the targeted samples. $\mathrm{R}^2$ identifies which combination of attributes is nominated for clustering, while the DB metric assesses the internal validation for these attributes or their individual forms if $C_s$ is empty. This analysis is performed separately for each of the selected clustering methods.

- *Detection phase*: Finally, upon completing the previous phase, the algorithm detects the optimal attribute and clustering method based on the DB scores. It performs the detection by returning the attribute subsets $k_{max}$ and the clustering method $m$ associated with the highest DB score 13. Essentially, the algorithm selects the clustering methods and attributes that yield the maximum DB score, which we have seen is likely producing better precision values.

## 3.3 MICRAST: An Approach for More Efficient Forecasting

In this section, we cover investigation tasks that lead to our new forecasting approach. This approach aimed at providing more efficient micro-prediction of clouds detailed patterns, as depicted in Figure 1.1. MICRAST overcomes the challenging characteristic of the cloud users' records, which suffers from sudden changes in their requests as illustrated in Figure 2.2. Such characteristics are not readily predictable by the models in the related works.

MICRAST offers an outline that enables micro-prediction through extracting segments of detailed patterns from cloud traces using clustering (as instructed by SeQual [99] and EFection [100]). This was conducted upon proving that the cloud traces demonstrate a sense of stability as depicted in the analysis investigation in Section 2.1.3, page 27, which aligns with the requirements of most forecasting models. This is followed by performing a comparison test between statistical and ANN-based forecasting models to select the best among them for our approach. We finalise this section by giving a thorough description of our proposed approach.

### 3.3.1 Forecasting Model Comparison

It's essential for developing an efficient forecasting approach to carefully select its model. Therefore, we carried out a comparison evaluation between various models listed in Table 2.8 to select the one that shows the highest performance in predicting cloud traces.

#### Setup Configuration

We set up the number of input layers based on the formula: (Number of attributes $\times$ window size). The window size represents the segments of the targeted traces that are selected for the forecasting model to capture during the learning process, as illustrated previously in Subsection 2.1.3. In the hidden layer, the desired model is selected (either RNN, LSTM, or GRU) with 100 units, an arbitrary but consistent choice across all models. Choosing 100 units ensures a balance between the complexity of the model and computational efficiency. This number is sufficient to capture intricate patterns within the cloud trace attributes without causing overfitting or incurring high computational costs. Such a configuration enables the network to learn the necessary patterns within the time series data. However, adjusting the number of units for each trace individually could potentially improve performance and is worth exploring in future work.

This is followed by the configuration of activation functions. These functions are essential elements in a neural network, as they determine the activation status of the corresponding neuron. Accordingly, we selected *tanh* for learning and *Hard sigmoid* for the recurrent layer, as prior research has indicated that these functions typically result in higher performance [101]. Finally, the hidden layer is connected to the third layer (the output layer). In this layer, the network is structured with one unit (output), and the ReLU activation function is selected to handle the output of the recurrent process. We present the implementation of the RNN network in the KNIME toolkit in Figure 3.4 as an example of the above configuration.

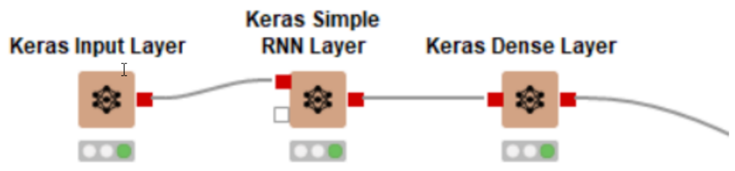


Figure 3.4: RNN configuration in KNIME

**Experimental Implementation**

As mentioned previously, we are developing this approach for both uni-attribute and multi-attribute forecasting scenarios to ensure broad applicability. To this end, we conducted two experiments—one for each scenario—to compare the performance of the forecasting models listed in Table 2.12 in predicting detailed usage patterns.

For this purpose, we selected the attributes Used Memory and Requested Number of Processors, as they represent user requests (i.e., consumption) from the cloud service provider. These attributes are widely available and generally applicable for forecasting across most traces. In contrast, other attributes, such as Requested Memory, were deemed unsuitable, as they exhibit a large proportion of constant values in many traces, as noted in [99] and [100], which limits their usefulness in providing meaningful information.

In the uni-attribute forecasting scenario, we focused on predicting patterns in the Requested Number of Processors by training the model on its own historical records. In the multi-attribute scenario, we repeated the process, this time including an additional input: the historical records of Run Time. This attribute was chosen because it shows a strong correlation with the Requested Number of Processors and is consistently available across all traces, making it a suitable candidate. This experiment tested the models' ability to capture inter-attribute dependencies when predicting a target value.

Both experimental scenarios were applied to all traces listed in Table 2.12. At this stage, input data were prepared manually without an automated preprocessing pipeline. We clustered the selected attributes to extract patterns suitable for comparison and set the forecasting window size to five, as smaller windows (fewer than five) were found insufficient for capturing recurring user behaviour in cloud traces.

In the ANN-based models, this configuration translates into five input neurons and one output neuron with an activating sequence return. Thus, five previous time steps of the selected attribute are used as input in the uni-attribute scenario. In contrast, for the multi-attribute scenario, the input neurons are doubled to ten to accommodate both attributes.

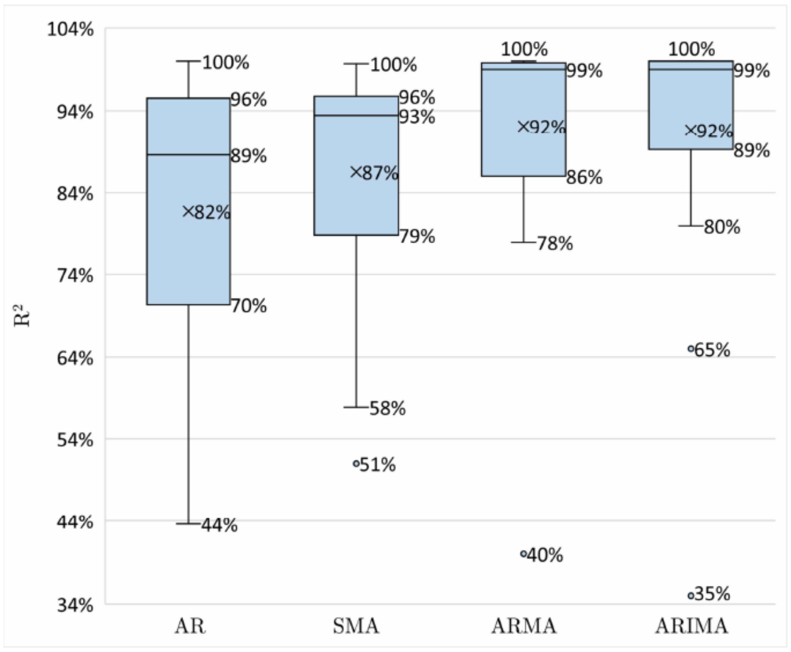
To implement these experiments, we split the prepared data into two subsets: 70% for training and 30% for testing. To evaluate the accuracy of the forecasting, we used the metrics MAPE and $R^2$, which measure how closely the predicted values align with the actual ones. Finally, we present boxplots illustrating the distribution range of $R^2$ scores for these models, providing insight into their accuracy and consistency. The results of the experiments are presented as follows:

**Uni-attribute Forecasting** In this experiment, we selected the models suitable for uni-attribute forecasting, as listed in Table 2.8. Figure 3.5 illustrates the boxplots showing the distribution ranges of $R^2$ scores for both the ANN-based and statistical models.

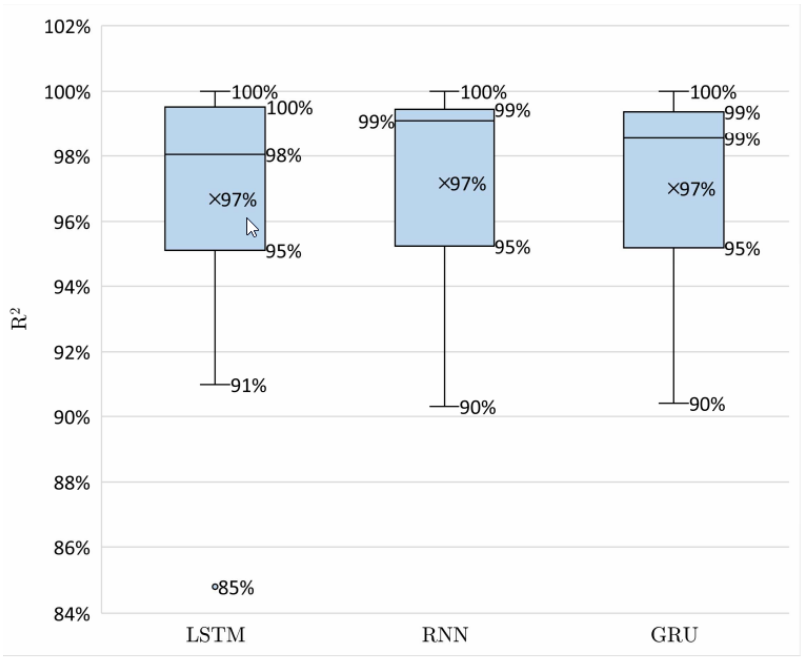
Figure 3.5a shows that the basic statistical models (i.e., AR and SMA) exhibit lower medians and wider ranges in their $R^2$ distributions compared to the more advanced models (i.e., ARMA and ARIMA). This is particularly evident in the AR model, where the lower whisker extends down to 44%. This performance degradation was mainly due to the ANL-Interpad and METACENTRUM-2009 traces, resulting in an average MAPE of approximately 2.1%. The SMA model demonstrated a slightly higher median and narrower distribution, with a better lower whisker at 58%, as observed in the DAS2 trace.

However, the SMA model is affected by an outlier at 51% in the HPC2N trace, and it exhibited a higher average MAPE of around 4%. In contrast, both ARMA and ARIMA models produced comparatively higher $R^2$ values and narrower distribution ranges, indicating that these more sophisticated models offer improved precision and consistency over basic statistical approaches. Nonetheless, both ARMA and ARIMA suffered from lower whiskers below 81% and outliers falling under 50%, primarily due to the CEA Curie trace. These issues contributed to a higher average MAPE of around 12% for both models. The root cause of the inconsistent performance of both basic and advanced statistical models is the nonlinear nature of users' consumption patterns in the affected traces. Consequently, linear autoregressive and statistical models, even in their more advanced forms, struggle to accurately forecast such behaviours. This highlights the uncertainty and limitations of using these models to predict cloud users' patterns.

In contrast, Figure 3.5b illustrates that the ANN-based models exhibit more stable performance in terms of $R^2$, with higher medians and narrower interquartile ranges. All three models(LSTM, RNN, and GRU) achieved similarly high and compact $R^2$ ranges, with the exception of a single outlier



(a) Statistical-based Models



(b) ANN-based Models

Figure 3.5: Boxplots of $R^2$ scores for uni-attribute forecasting models

at 85% for the LSTM model, corresponding to the LLNL ATLAS trace. This outlier increased LSTM's average MAPE to approximately 0.7, compared to an average of 0.4 for both RNN and GRU.

In conclusion, these results indicate that, for uni-attribute forecasting, ANN-based models, particularly RNN and GRU, are more suitable for predicting cloud users' patterns, offering greater consistency and adaptability to non-linear usage patterns.

**Multi-attribute Forecasting** In this experiment, we utilised the models applicable to multi-attribute forecasting, as listed in Table 2.12. The boxplot in Figure 3.6 illustrates the performance of both statistical and ANN-based models. As shown, the VAR model exhibited a lower median, a wider range of $R^2$ values, and a greater number of outliers compared to the ANN-based models. Specifically, the outliers accounted for 72% in the forecasting of the DAS2 trace, 67% for SDSC BLUE, and 58% for CEA Curie.

As mentioned previously, the attributes in these traces exhibit non-linear characteristics. Therefore, these results indicate that the simple autoregressive process of the VAR model is unable to effectively capture the correlations between such attributes. Instead, it tends to interpret the complex patterns in these attributes as noise, leading to a relatively high average MAPE of approximately 3.57%.

On the other hand, Figure 3.6 shows that the ANN-based models handled these challenges with better overall performance, achieving an average MAPE of approximately 1.9%. These models were able to effectively capture the relationships between attributes to predict the target values. However, both the LSTM and GRU models exhibited an outlier with an $R^2$ of 58% for the OSC Cluster trace. Notably, this trace achieved 95% $R^2$ in the uni-attribute forecasting scenario using the same models.

This decline in performance can be attributed to overfitting caused by the inclusion of an additional attribute, Run Time, alongside Requested Number of Processors. The overfitting issue was particularly evident in the DAS2, SDSC BLUE, and CEA Curie traces. The long-term memory capabilities of the LSTM and GRU models can lead to such overfitting when attempting to learn complex relationships between attributes characterised by abrupt changes.

In contrast, the RNN model, due to its simpler memory structure, demonstrated more stable performance and produced a narrower range in the boxplot. Compared to the other ANN-based and statistical models, the RNN model achieved superior forecasting accuracy: 96% for OSC Cluster, 96% for DAS2, 93% for SDSC BLUE, and 90% for CEA Curie. These results suggest that the RNN model effectively manages noisy patterns and mitigates overfitting, while maintaining high predictive accuracy.

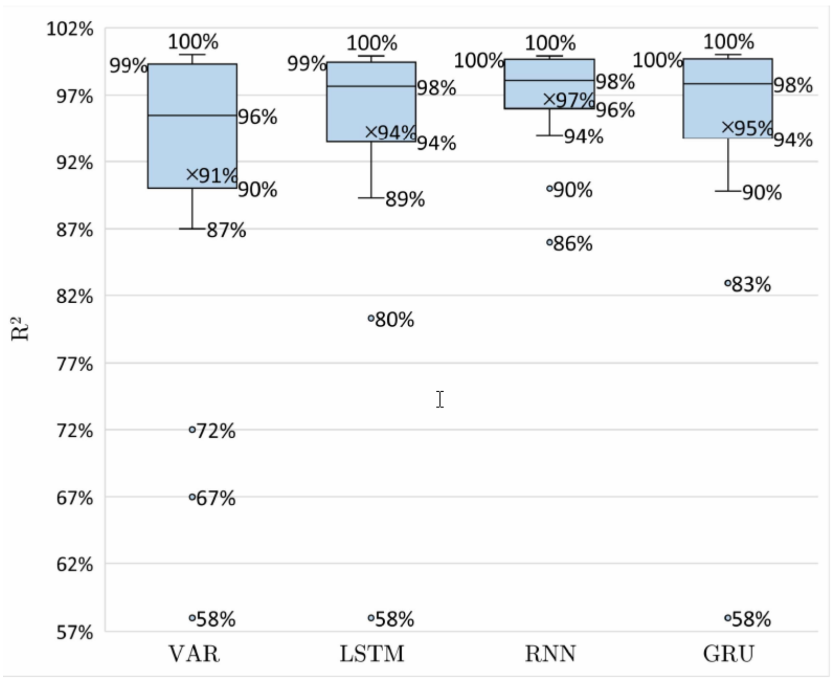


Figure 3.6: Boxplots of $R^2$ scores for multi-attributes forecasting models

**Findings** Among the evaluated models, the RNN consistently achieved high accuracy across both uni-attribute and multi-attribute forecasting scenarios, recording an average $R^2$ of approximately 97%. Moreover, it maintained this level of performance even on traces that posed significant challenges. These results make the RNN model a strong candidate for our approach. The detailed architecture of MICRAST and its integrated RNN network is presented in the subsequent subsections.

### 3.3.2 The Proposed Approach: MICRAST

We propose the MICRAST approach to predict the future consumption patterns of cloud users. Our approach achieves this through a pipeline comprising segmentation, pre-processing, and forecasting, as illustrated in Figure 3.7. In this section, we cover both the training and forecasting phases of the approach, in comparison to existing methods shown in Figure 3.8.

**Training Phase** This phase proceeds as follows:

- The extraction pipeline employs clustering to uncover, from the input trace, the hidden patterns that drive users' requests. Based on our findings in [102], clustering has demonstrated a strong capability for such extraction. To ensure efficient clustering, we perform two main tasks. First, we filter the trace by excluding attributes that hold the same value for more than 80% of the records. Such attributes are considered unsuitable for clustering, as illustrated in [99].

  Second, we apply two tools: the Sequential method of clustering Quality (SeQual) and the Effectiveness detection of clustering quality (EFection), to address both uni-attribute and multi-attribute feature selection scenarios. The SeQual method ranks individual attributes to determine the best candidate for extraction when the user opts for uni-attribute forecasting. Conversely, the EFection technique selects the most compatible combination of attributes for extraction in the multi-attribute forecasting scenario. Notably, if EFection selects only one attribute, it is recommended that the user opts for uni-attribute forecasting instead. Additionally, we use EFection to select the most suitable clustering method for the chosen attributes. The selected clustering method then groups similar historical usage records along with their submission times to form consumption patterns for each user (see Table 5). The output of this task is a set of segments representing detailed patterns.

- Parallel pre-processing pipelines prepare each segment of detailed patterns for prediction. In their clustered form, these segments exhibit non-uniform scales and formats that do not satisfy the requirements for effective data forecasting, as presented in Section 2.1.3 (page 25). Therefore, in these pipelines, we perform uniforming processes in parallel, separately for each segment, as depicted in Figure 3.7. First, the current time sequence for each segment is standardised into a consistent format across all traces. We also apply time alignment to synchronise these segments on the same time scales. Second, linear interpolation is implemented to address any missing records.

  Subsequently, the data in each segment are normalised to a range between 0 and 1. This normalisation is essential for efficient forecasting, as cloud workload traces often exhibit values on vastly different scales. Without normalisation, such disparities can hinder forecasting performance, whereas normalisation facilitates better compatibility with ANN forecasting models. The output of these pipelines is a set of uniform segments, each ready to be used as input for forecasting training.

- A parallel forecasting pipeline feeds the uniformed segments into the RNN model for training. It is important to emphasise that the RNN model is configured according to the setup presented in Section 3.3.1, which demonstrated high performance in our comparative experiments. Once the RNN model is sufficiently trained, this pipeline produces a trained network for each segment, which is then stored and used later to forecast new input traces from the service provider's system.

  In addition, this pipeline also computes the average centroid for each segment. These centroids are stored alongside the corresponding trained networks to facilitate the prediction phase.

**Prediction Phase** In this phase, our approach follows the same segmentation and forecasting pipeline used in the training phase. As shown in Figure 3.7, the new input data are first clustered into segments of detailed patterns, after which the average centroid is calculated for each segment. These segments are then fed into the appropriate trained networks to predict future events. This is achieved by comparing the centroid of each new segment to the stored centroids from the training phase. When a match is found within a defined range, the corresponding trained network is selected and applied to the current segment for prediction.

In contrast to our approach, the methods adopted in recent cloud studies (as shown in Table 3.2) follow a singular prediction pipeline. As illustrated in Figure 3.8, these methods are designed to perform data preparation tasks, followed by forecasting, all on the full input dataset without considering any detailed segmentation. The prepared data are used to train a single forecasting model, which is then directly applied to the entire new input data during the prediction phase. This traditional pipeline is referred to as *Macro-prediction*. A detailed comparison of these approaches, discussed earlier in Section 2.3, is presented in Table 3.2. The table outlines their prediction focus, level of detail, methodology, and forecasting scope. Unlike these methods, which operate at a macro level, MICRAST adopts a micro-level strategy tailored to individual user behaviour, offering more fine-grained and personalised predictions.

make the following table algin with the textwidth :

Table 3.2: Comparison of MICRAST with Recent Forecasting Approaches

| **Approach** | **Prediction Focus** | **Detail Level** | **Methodology** | **Level** |
|---|---|---|---|---|
| **MICRAST** | User consumption prediction | High (user-level) | Pre-processing + LSTM-RNN | **Micro** |
| **VTGAN** [91] | Workload values and trends | Low (workload-level) | GAN-based non-linear model | **Macro** |
| **F.Prophet** [103] | VM workload behaviour | Low (workload-level) | Prophet with tuning and pre-processing | |
| **LSTM** [96] | SLA-aware workload prediction | Low (workload-level) | LSTM for time-series forecasting | |
| **WGAN-gp** [92] | Dynamic workload prediction | Low (workload-level) | Transformer + Wasserstein GAN | |
| **CNN-LSTM** [104] | Cloud workload prediction | Medium (system-level) | CNN + LSTM hybrid model | |
| **MAG-D** [21] | CPU and memory usage | Medium (resource-level) | GRU on data centre traces | |
| **RVLBPNN + K-means** [90] | Response time-based trends | Medium (workload-level) | RVLBPNN + clustering | |
| **esDNN** [105] | Workload & resource optimisation | Medium (system-level) | GRU for time-series prediction | |
| **LLM** [97] | Power demand prediction | Medium (task-level) | LLM with hallucination control | |
| **LSTM/RNN** [93] | Resource optimisation | Medium (system-level) | LSTM/RNN on past workload samples | |
| **RVLBPNN** [22] | Workload trend forecasting | High (workload-level) | Neural network with latency awareness | |
| **Self LSTM** [94] | Cloud demand prediction | Medium (system-level) | LSTM + self-directed learning | |

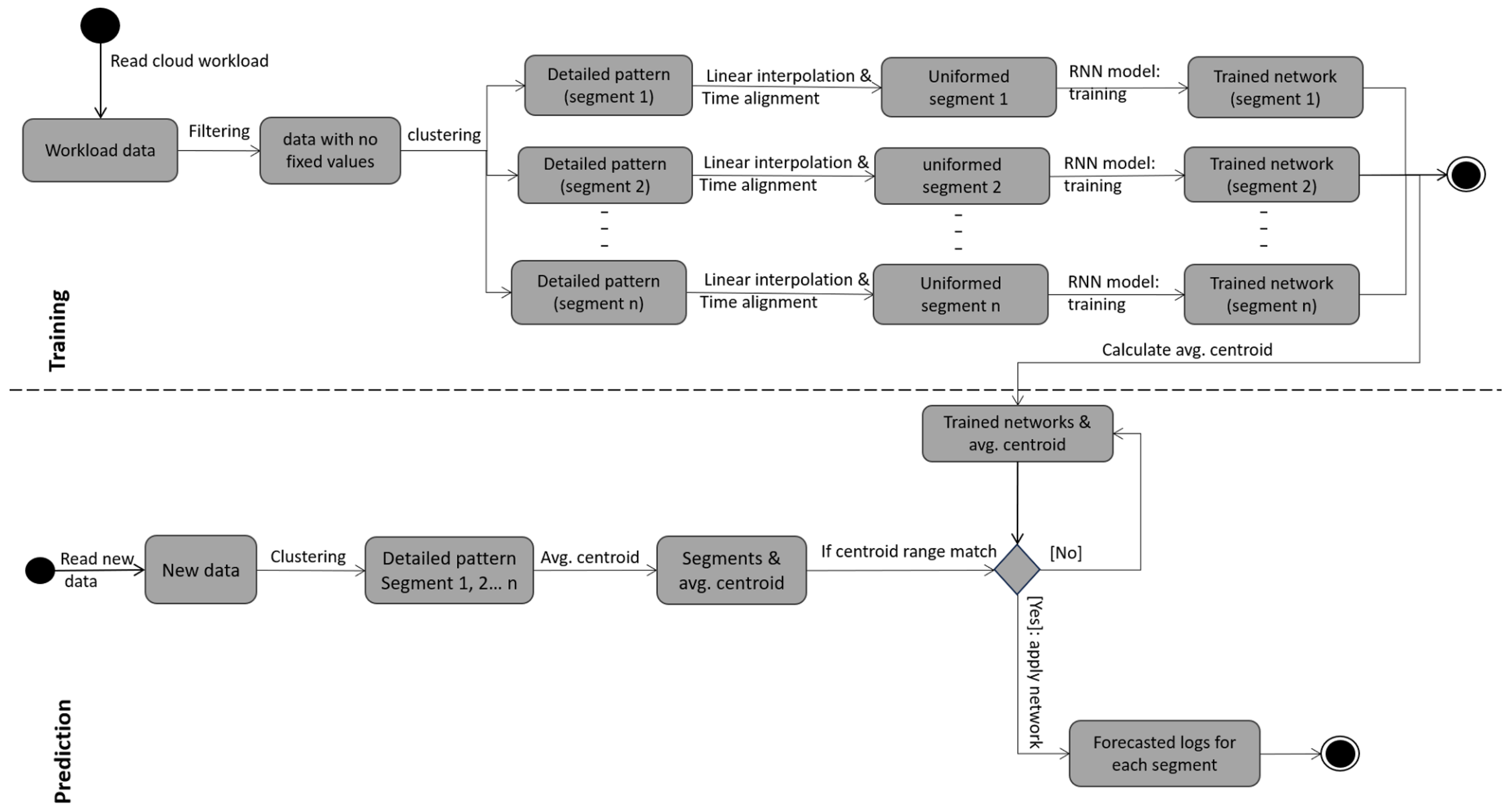

Training
Read cloud workload
Workload data
Filtering
data with no fixed values
clustering
Detailed pattern (segment 1)
Linear interpolation & Time alignment
Uniformed segment 1
RNN model: training
Trained network (segment 1)
Detailed pattern (segment 2)
Linear interpolation & Time alignment
uniformed segment 2
RNN model: training
Trained network (segment 2)
Detailed pattern (segment n)
Linear interpolation & Time alignment
Uniformed segment n
RNN model: training
Trained network (segment n)
Calculate avg. centroid
Prediction
Trained networks & avg. centroid
Read new data
New data
Clustering
Detailed pattern Segment 1, 2… n
Avg. centroid
Segments & avg. centroid
If centroid range match
[No]
[Yes]: apply network
Forecasted logs for each segment

Figure 3.7: The MICRAST approach

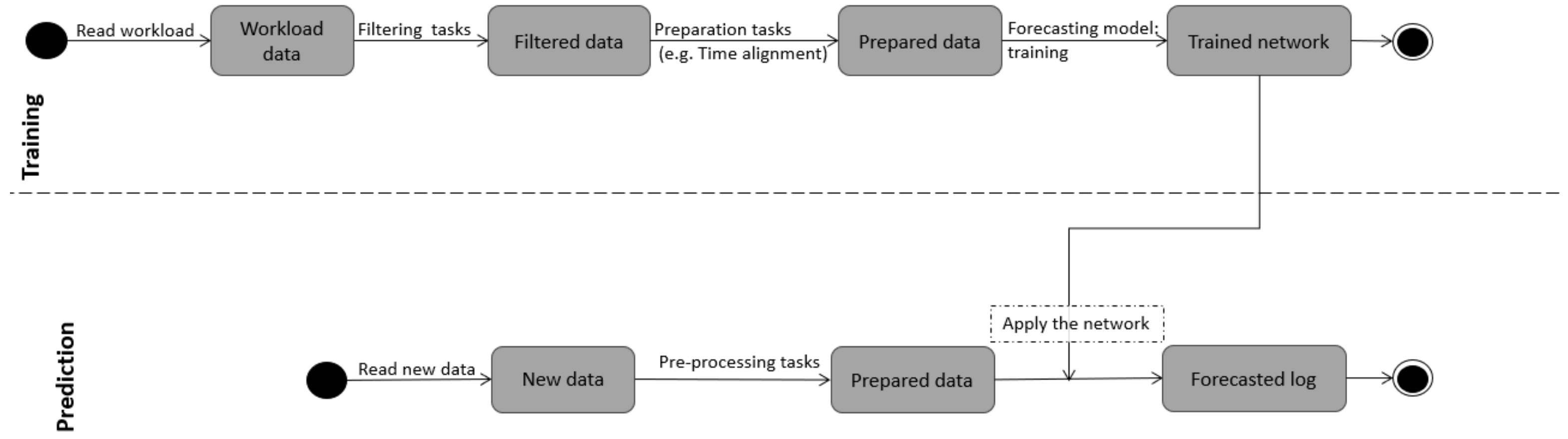


Figure 3.8: The macro-prediction approaches

### 3.3.3 Summary

This chapter presented the methodology underlying the proposed tools designed to enhance the extraction and forecasting of detailed patterns in cloud traces. We first introduced the development of SeQual, a novel unsupervised feature selection method, along with the observations that motivated its design. This method ranks the attributes used in the clustering-based extraction process by leveraging the SC metric to produce a sequential quality measure.

Subsequently, we described the EFection technique and the preliminary comparative experiments conducted to determine its internal validation metric. This technique identifies the most effective combination of attributes and clustering method without requiring analysis of the full dataset. It achieves this by employing both the DB index and the $R^2$ metric.

Finally, the chapter concluded with an explanation of our forecasting approach, MICRAST, which aims to enable more accurate micro-prediction of cloud users' behavioural patterns. This approach integrates the aforementioned extraction tools to uncover hidden user patterns in cloud traces and predict them with greater precision.

# Chapter 4

# Evaluation and Results

This chapter presents the evaluation and results of our proposed tools. The evaluation was conducted through two main tasks: extraction tools validation and forecasting approach validation, as illustrated in Figure 4.1.

We begin by selecting the traces based on two main criteria: being supervised and containing applicable attributes, as shown in Table 2.4. For the extraction tools validation, we start by preparing the data through disregarding those attributes that contain labels (i.e., User IDs). Later, we use these attributes as ground truth for validating the quality of extraction, as illustrated in Section 2.2.2.

To validate SeQual, we carry out a comparative evaluation against commonly used feature selection techniques. This is conducted to demonstrate the ability of our method to perform better without the need for supervisory input, which is required in other related techniques. To evaluate EFection, we apply three core experiments. These include measuring our technique's accuracy across different selection scenarios and assessing its applicability in two clustering-based cloud studies. Finally, we compare the performance of EFection with two well-established tools used for dimensionality reduction and clustering method selection. These experiments aim to cover all aspects of our technique's usability. Both tools' validation followed the evaluation process described in Section 2.2.2, where clustering was employed to assess the quality of extraction by identifying relevant user trace labels.

For the forecasting approach, we also begin by preparing the data, selecting attributes that represent users' requests for prediction. These selected attributes are used as inputs for forecasting, as our focus in this thesis is on capturing and predicting consumption patterns. To validate MICRAST's performance, we compare it against a case study from related literature, applying both uni-attribute and multi-attribute forecasting for generality. This is followed by evaluating its range of confidence by testing its prediction accuracy across different scales of future events. These experiments are designed to examine the robustness and applicability of our forecasting method across different forecasting types and data conditions.

The following sections detail these evaluations and their corresponding results.

## 4.1 Validation of the Extraction Tools

### 4.1.1 A Comparative Evaluation of SeQual

As illustrated in Figure 4.1, we begin with a comparative evaluation to assess the performance of the newly proposed SeQual method. This was carried out using 10,000 records randomly selected from the supervised trace attributes listed in Table 2.4. As described in Chapter 3, Section 3.1, the purpose of the SeQual method is to rank and select clustering attributes to extract detailed consumption patterns from cloud workload traces. Therefore, the evaluation criterion is based on each attribute's ability to produce meaningful patterns that can distinguish user labels via clustering, consistent with the process described in Section 2.2.2.

To perform the comparison, we employed a ranking-difference approach. Each attribute was individually clustered in an attempt to reveal the corresponding users' labels. The clustering outputs were then evaluated against the user ID attribute using three quality metrics: Precision, Entropy, and Adjusted Rand Index. The closer the clustering results are to the actual user IDs, the higher the true rank of the attribute.

Next, we applied the SeQual method to rank the attributes and compared its output with three commonly used feature selection methods: Laplacian Score (LS), Random Forest (RF), and Principal Component Filtering (PCF), as introduced in Section 2.1.2. For this comparison, we used all the unsupervised traces listed in Table 2.5.

For the supervised methods (PCF and RF), the user ID was used as the reference for generating the ranking. In contrast, for the unsupervised methods (SeQual and LS), the input attributes were evaluated without exposing any direct user identifiers (e.g., user IDs, user groups, or executable names). The effectiveness of each selection method was measured using the three metrics individually to avoid introducing bias from reliance on a single quality measure.

The overall comparison of SeQual with LS, PCF, and RF was performed using a weighting process. In this process, each attribute selected by a given method was assigned a weight based on the proximity of its clustering quality

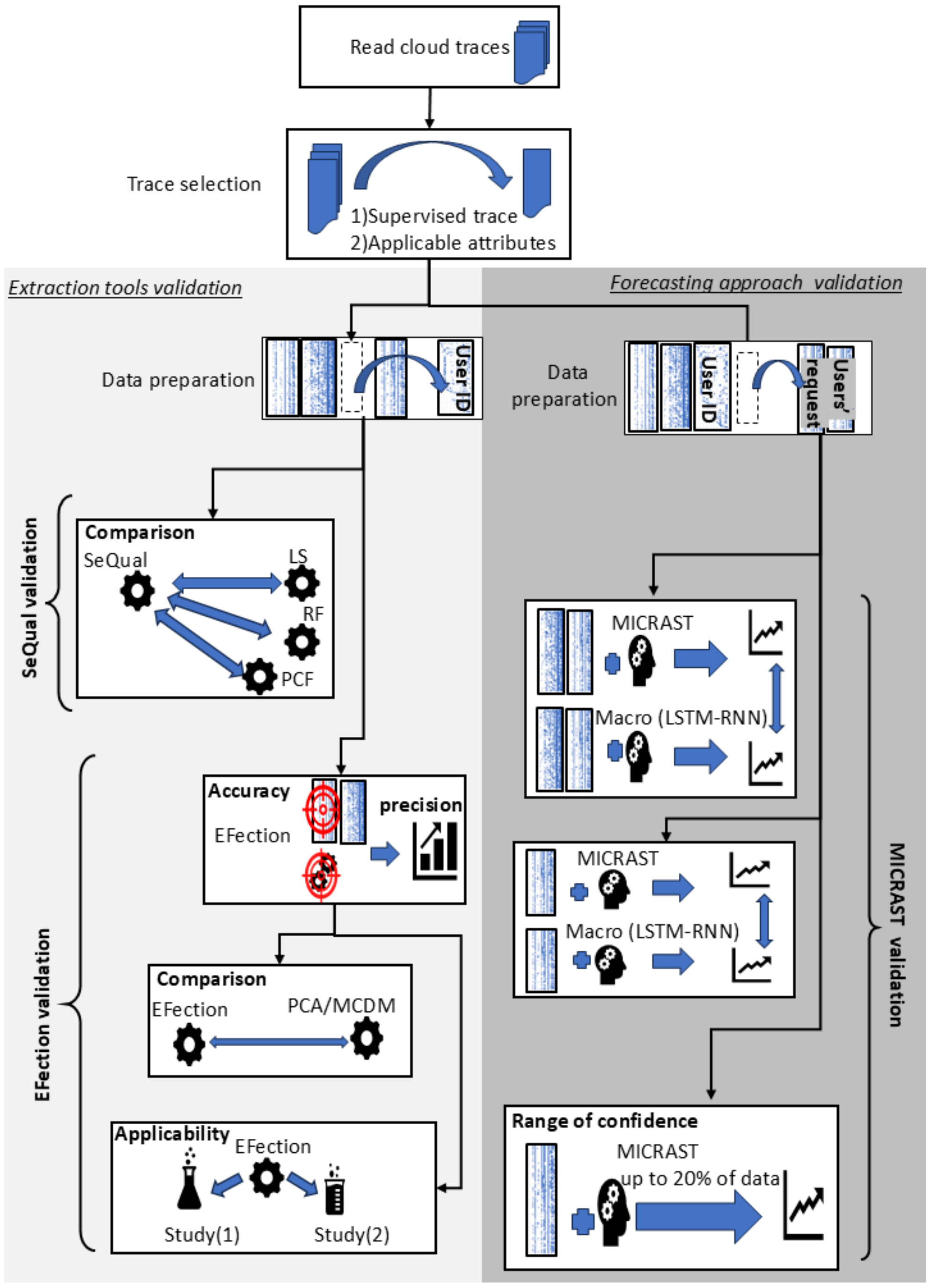


Figure 4.1: Outline for validating the proposed tools

to the best-performing attribute, calculated as:

$$\text{Weighted selection} = \frac{\text{Clustering quality for selected attribute}}{\text{Highest clustering quality}}$$

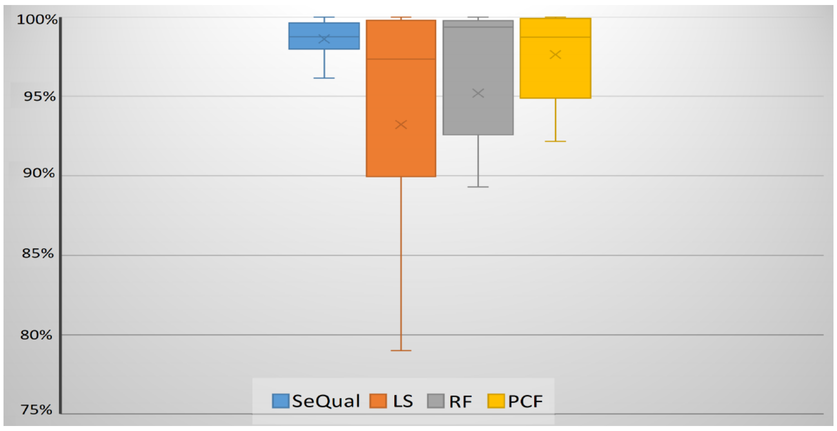
In addition, we extended the evaluation by analysing the distributional characteristics of the supervised traces used for attribute ranking. Specifically, we examined two statistical measures relevant to clustering behaviour: the Coefficient of Variation (CV) and Skewness. We computed these measures for all attributes across the 19 supervised traces, then identified the characteristic ranges within which attribute rankings were most effective. This analysis indicates a potential correlation between the quality of attribute ranking and these statistical properties. Accordingly, unsupervised traces that fall within similar ranges of CV and Skewness are expected to exhibit ranking performance comparable to those observed in the supervised traces.

**Testing and Results**

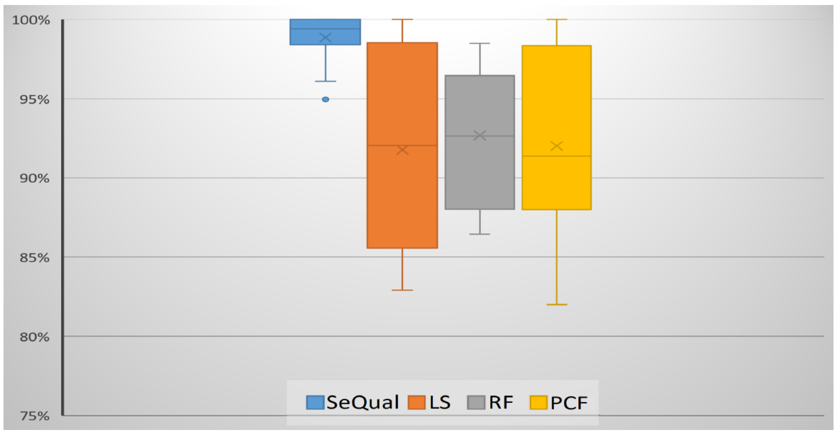
To present the performance of each method in greater detail, we displayed the ranges for all 19 traces listed in Table 2.4, along with all performance metrics, in the boxplots of Figure 4.2. These boxplots illustrate the distribution of each feature selection method's performance. A method exhibiting the highest and least variable range of clustering quality across all validation metrics is considered to demonstrate more stable performance. Accordingly, SeQual showed the greatest stability among the methods tested.

As shown in Figure 4.2, the SeQual method outperforms all other evaluated methods across the three quality metrics: Precision, Entropy, and Adjusted Rand Index. Specifically, according to the Adjusted Rand Index, SeQual achieves an attribute ranking accuracy of approximately 90%, compared to 74% for both LS and RF, and 79% for PCF. Regarding the Precision metric, SeQual attained a score of 99%, while the other methods achieved around 92%. Conversely, the Entropy metric exhibited less variation among methods, with SeQual scoring 99% and LS, RF, and PCF all achieving roughly 98%.

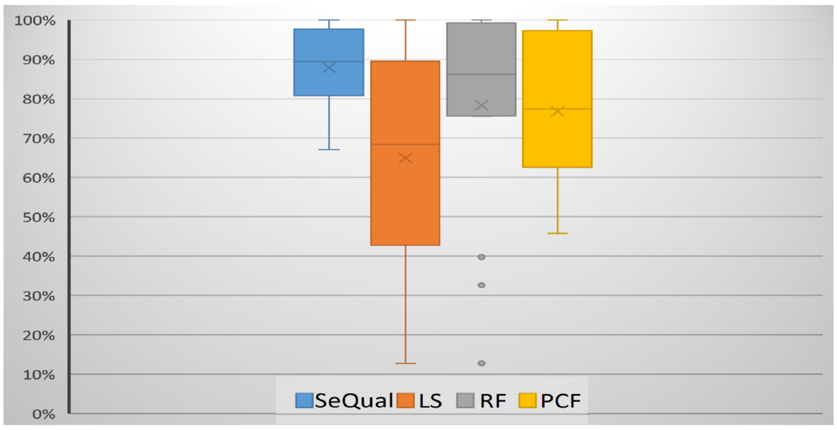
Based on these results, it can be concluded that despite being an unsupervised approach, SeQual demonstrates superior performance over the compared methods by margins ranging from approximately 8% to 28%. This improved performance is likely attributable to the fact that SeQual directly performs sample clustering during the ranking process, thereby capturing the underlying data structure more effectively.



(a) Performance of methods based on Entropy



(b) Performance of methods based on Precision

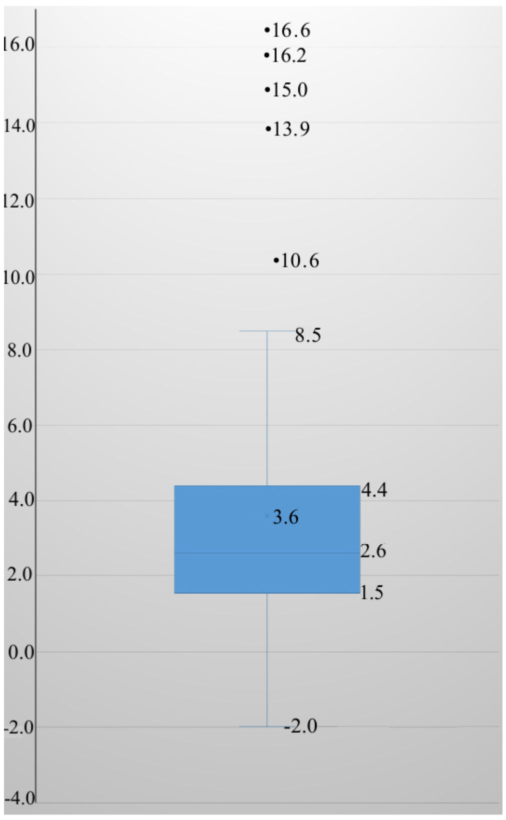


(c) Performance of methods based on Adjusted Rand Index

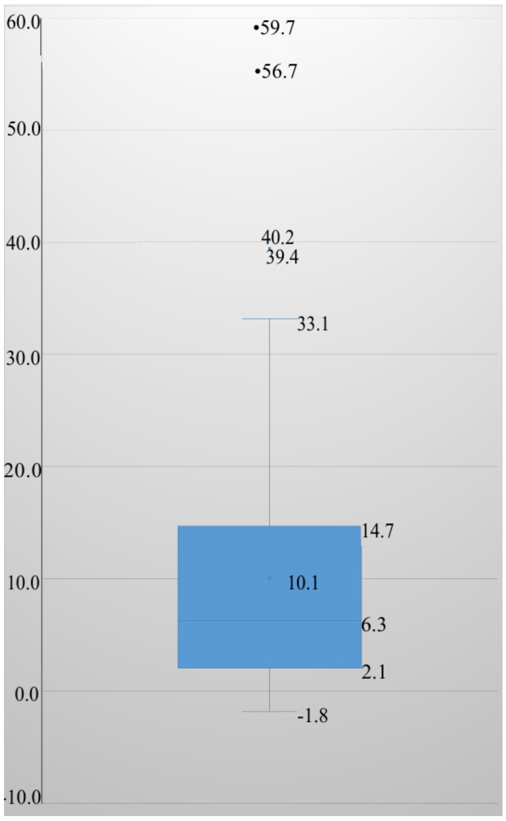
Figure 4.2: Boxplots showing the distribution of methods' performance

### Compatibility of Unsupervised Traces

Finally, we assessed the compatibility of unsupervised traces for ranking by the SeQual method through an analysis of the ranges of the Coefficient of Variation (CV) and Skewness for both supervised and unsupervised traces. It is expected that traces whose CV and Skewness fall within similar ranges as those of supervised traces will be ranked by SeQual with comparable accuracy. Figure 4.3 shows boxplots of the CV and Skewness distributions for the relevant attributes across all 19 supervised traces. By comparing these distributions to the clustering quality ranges for supervised traces, it was observed that traces without out-of-range characteristics (in either CV or Skewness) have approximately an 80% chance of being ranked with high accuracy by SeQual when using Precision as the accuracy measure. Here, *out-of-range* refers to outliers beyond 33.1 and below -1.8 in the Skewness boxplot, and beyond 8.5 and below -2.0 in the CV boxplot (see Figure 4.3).



(a) CV distribution



(b) Skewness distribution

Figure 4.3: Ranges of statistical distributions for supervised traces

Table 4.1: Range of ranking accuracy for each quality metric

| Quality metric | Level of out-of-range | Quality range |
|---|---|---|
| adjusted rand index | 0% | 98% to 100% |
| Precision | 0% to 20% | |
| Entropy | 0% to 30% | |
| adjusted rand index | 0% to 20% | 85% to 95% |
| Precision | 20% to 30% | 96% to 98% |
| adjusted rand index | 20% to 30% | 75% to 85% |

Table 4.2: Range of attribute ranking accuracy across traces and metrics

| Company | unsupervised Out of Range | Adj. Rand Index | Entropy | Precision |
|---|---|---|---|---|
| Bitbrains | 18% | 85% to 95% | 98% to 100% | 98% to 100% |
| Materna | 0% | 98% to 100% | 98% to 100% | 98% to 100% |
| Ali Baba | 0% | 98% to 100% | 98% to 100% | 98% to 100% |
| Facebook | 60% | not recommended | not recommended | not recommended |
| Google-vm-reading | 0% | 98% to 100% | 98% to 100% | 98% to 100% |
| GWAT-AuverGrid | 0% | 98% to 100% | 98% to 100% | 98% to 100% |

Based on the observations above, we calculated the out-of-range level of the targeted traces using the following equation:

$$\text{level}_{\text{out of range}} = \frac{50 \times \#\text{out of range}}{\#\text{attributes}} \tag{4.1}$$

Consequently, we computed the out-of-range level for all 19 supervised traces, as shown in Figure 4.4. The cleanest traces exhibit 0% out-of-range levels. In contrast, noisy traces, defined as those with all their attributes outside the specified CV and skewness ranges, exhibit an out-of-range level of 100%. Based on this, we formulated a range of ranking accuracy for each quality metric, as shown in Table 4.1. It is worth noting that, based on Entropy, we expect high quality across a wider out-of-level range compared to Precision and the Adjusted Rand Index. For the same level of out-of-range, we observe varying quality ranges for Precision, and even more so for the Adjusted Rand Index. This explains why Entropy does not appear in the second and third rows of Table 4.1.

In the next stage of this evaluation, we applied the above observations to the unsupervised traces listed in Table 2.5. Table 4.2 presents the calculated out-of-range level for each unsupervised trace, along with the expected quality scale of attribute ranking using the SeQual method. According to Table 4.2, the traces from Materna, Ali Baba 2007, GWAT AuverGrid, and Google VM Reading exhibit a 0% out-of-range level. This indicates that these traces are expected to yield high ranking accuracy when using the SeQual method. In contrast, the statistical characteristics of the Bitbrains trace show an 18% out-of-range level, suggesting an expected ranking accuracy between 85% and 95% according to the Adjusted Rand Index, and between 98% and 100% based on both the Entropy and Precision quality metrics. Additionally, the Facebook Hadoop trace shows a 60% level of noise. According to our analysis, such a high out-of-range level makes this trace unsuitable for attribute ranking using the SeQual method.

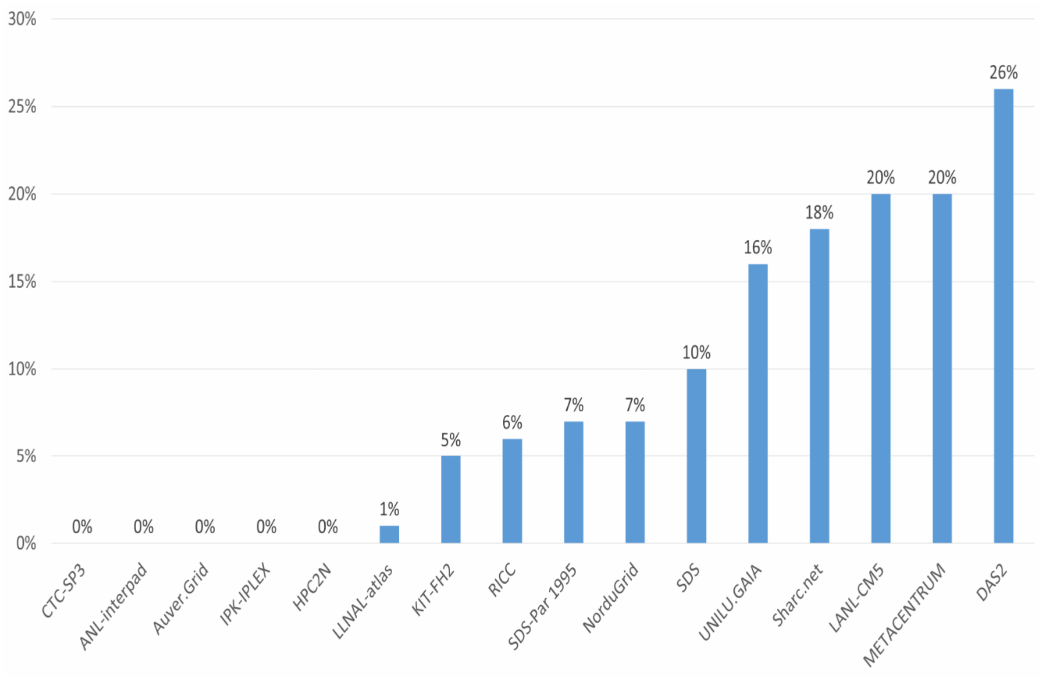


Figure 4.4: level of out-of-range attributes in workload traces

**Finding and Discussion**

The evaluation of the SeQual method was carried out by testing its ability to rank attributes across 19 supervised traces. These traces were treated in an unsupervised manner, and the results were then compared against their known supervised information (i.e., users' labels). This was followed by a comparative assessment between the new SeQual method and widely used feature selection techniques, including supervised methods (RF and PCF) and an unsupervised method (LS). The SeQual method demonstrated higher accuracy and greater stability than these alternatives, based on the quality metrics of Precision, Entropy, and Adjusted Rand Index. Notably, this performance was achieved without full trace analysis or the need to predefine supervisory parameters such as the number of clusters or expected classification labels.

In addition, we evaluated the method's suitability for ranking attributes in unsupervised traces. This was achieved by analysing the statistical characteristics of traces that are recommended for ranking with our method. The results showed that the majority of these traces can be ranked with high expected accuracy using the SeQual technique.

### 4.1.2 EFection Validation: Accuracy, Comparison, and Applicability

In this section, we evaluated the EFection technique through three main experiments, as shown in Figure 4.1. First, we measured the accuracy of the technique when applied to detect combinations of attributes and clustering methods for cloud workload traces. This accuracy assessment provides insight into how well the technique performs across different cases.

Next, we conducted a comparative evaluation of EFection against the most recent related approaches in the literature. This evaluation aims to demonstrate and validate the superiority of our technique over existing methods. Finally, we assessed the applicability of EFection in identifying both suitable clustering methods and attribute sets in real-world workload studies.

Through these three experiments, we aim to comprehensively validate the EFection technique and demonstrate its capability in practical and theoretical scenarios.

Table 4.3: Scenario for correct EFection suggestion

| Clustering method | Attributes | MD clustering precision | Individual Clustering precision | EFection Suggestion |
|---|---|---|---|---|
| K-means | Wait Time | 77% | 65% | Run Time with F.First |
| | Run Time | | 81% | |
| F.First | Wait Time | 91% | 86% | |
| | Run Time | | 96% | |

**EFection Accuracy**

In the accuracy experiment, we used various combinations of clustering methods and attributes sampled from all the traces listed in Table 4.3. We began by measuring the precision of clustering for each of these samples. As in previous experiments described in this dissertation, clustering was performed with the goal of extracting users' labels based on their usage patterns, and precision was measured accordingly.

The attribute and method combination yielding the highest precision was considered the optimal choice. We then applied the EFection technique to predict the best attribute-method combination for each input. To evaluate its performance, we calculated the overall percentage of predictions that matched the optimal choices. For the incorrect suggestions, we measured the deviation from the optimal selections and used these to determine the percentage error.

Based on these measurements, we calculated the overall accuracy of the EFection technique. We illustrated this experiment using two representative scenarios, as outlined below:

**Scenario (1): Accuracy Measurement for Correct Detection** In this scenario, we considered two attributes (Run Time and Wait Time) from the PIK-IPLEX trace and two clustering methods (K-means and F. First). We used the EFection technique to identify the best combination of attributes and method for clustering.

First, we measured the precision of the full trace for these attributes. As shown in Table 4.3, the precision for clustering Wait Time was 65% and 81% for Run Time when using the K-means method. By combining these two attributes, the precision dropped to 77%. While using the F. First method, the Wait Time recorded 86% precision and 96% for the Run Time. Similarly, the results of their combination dropped to 91%. By comparing these results for the F. First and K-means methods, we noticed that F. First showed higher precision than K-means, with a better result for individual clustering. These results showed that it was better to use Run Time individually with the F. First method for the clustering process.

Second, we compared the above results with the EFection suggestion. Our technique suggested clustering the attribute of Run Time individually rather than combining it with Wait Time when using K-means. Similarly, for F. First, it proposed clustering run time individually. For method comparison, EFection suggested using the F. First clustering method rather than K-means. The comparison showed that the suggestion from our technique chose correctly the highest possible precision for the above scenario, in which the accuracy will be recorded as (96%/96%= 1) based on the following equation:

$$\text{Accuracy} = \frac{\text{suggested option precision}}{\text{highest possible precision}} \tag{4.2}$$

**Scenario(2): Accuracy measurement for wrong detection** This scenario considers two attributes of Run Time and Requested Time from the SDSC DS 2004 trace and both K-means and EM methods. The result in Table 4.4 showed that the precision was the highest, around 58%, for clustering Run Time individually with K-means. While EFection suggested using the attribute (Requested Time and Run Time) and the K-means method in the clustering process. In this scenario, we measured the difference between the precision of our technique's suggestion and the highest precision. Using the equation 4.1.2, the error percentage for this case recorded (55%/58%= 0.94).

**Experimental results** By applying the above evaluation methodology, the EFection technique was able to detect the best combination of attributes and clustering method with optimal choice (Accuracy = 1) in 83% of these cases. While the distribution of error percentages ranged between 2.8% and 10.8%, as shown in Figure 4.5.

### Comparison with Recent Related Works

As previously mentioned, we introduced EFection as an automated technique for simultaneously detecting useful combinations of attributes and clustering methods. Most related works address these factors separately, offering individual techniques for each. To address dimensionality, Daraghmeh et al. [78] employed a PCA-based approach, while Barak and Mokfi [14] utilised an

Table 4.4: Scenario for wrong EFection suggestion

| Clustering method | Attributes | MD clustering precision | Individual Clustering precision | EFection Suggestion |
|---|---|---|---|---|
| K-means | Run Time | 55% | 58% | combine (Requested Time and Run Time) with K-means method |
| | Requested Time | | 57% | |
| F.First | Run Time | 53% | 53% | |
| | Requested Time | | 55% | |

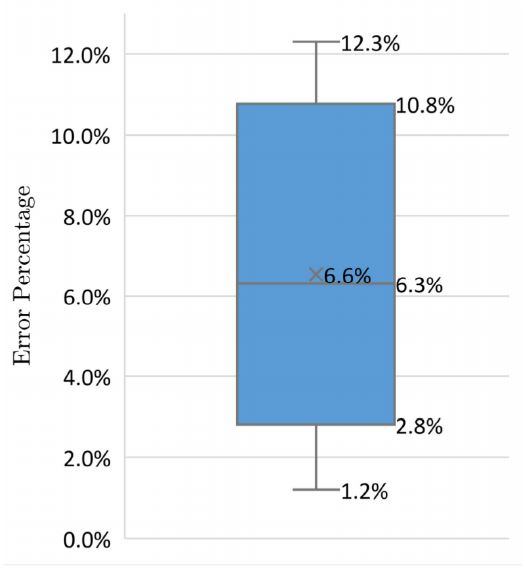


Figure 4.5: Distribution of error percentages for EFection accuracy

MCDM group methodology for method selection. Therefore, we evaluated EFection by comparing it with an integrated implementation of these two approaches (PCA & MCDM).

For the experiment, we used samples comprising clustering methods from Table 2.14 and attributes from Table 2.4. Regarding PCA & MCDM, we first applied the MCDM methodology to select the clustering method, followed by PCA for dimensionality reduction of the attributes. Then, we applied EFection on the same samples. Similar to other experiments in this disserta-

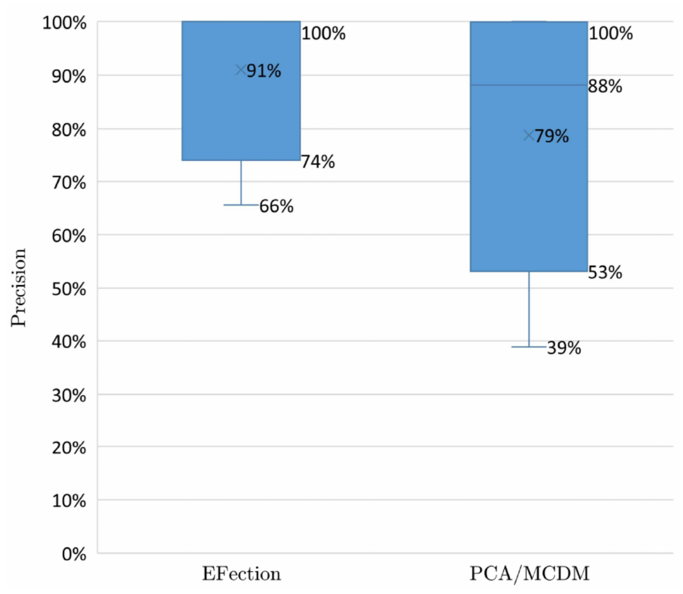


Figure 4.6: Comparison between EFection and PCA & MCDM performance

tion, we used the clustering process with the selected methods and attributes to extract users' labels and calculated the precision of the results. Finally, we compared the precision of the suggestions from EFection and PCA & MCDM with the combination that yielded the highest clustering precision, calculating how closely each suggestion matched the highest result.

The results, shown in Figure 4.6 as boxplots, reveal that EFection's precision ranged from 74% to 100%, with a median of 100%. Meanwhile, the PCA & MCDM approach achieved precision ranging from 53% to 100%, with a median of 88%. Additionally, EFection had an average precision of 91%, compared to around 79% for PCA & MCDM. This discrepancy is attributed to the PCA component generating new features, which reduced clustering precision. These findings demonstrate that EFection offers better precision and stability compared to the combined PCA & MCDM approach, improving accuracy by 11 percentage points.

### EFection Applicability in Clustering-based Studies

We evaluated EFection's applicability by comparing the clustering quality of its selections against the preferences in two related studies. We conducted this by employing our technique to select the best method for a utilization improvement study [7] and the most effective attributes for a pattern extraction study [70]. We selected these works as they align with the dataset criteria of this thesis, as presented in Table 2.4. Our technique's comparison

Table 4.5: Comparison between EFection's suggestions and attributes used in a utilisation improvement study

| Clustering method | CT | NT | Reference |
|---|---|---|---|
| K-means (Euclidean) | 85% | 15% | Yousif and Al-Dualimy [7] |
| K-means (Manhattan) | 82% | 18% | Yousif and Al-Dualimy [7] |
| Density Based | 66% | 34% | Yousif and Al-Dualimy [7] |
| SOM | 61% | 39% | EFection suggestion |

against each study is illustrated as follows:

**Compare Clustering Method Selection Against Utilisation Improvement Study** The first experiment compared the clustering quality of the method suggested by our technique with the existing results in [7]. In their study, the authors employed K-means and density-based methods to cluster 12,500 records of Google workload traces, aiming at improving resource utilisation. For distance measures in K-means clustering, this work used Euclidean and Manhattan metrics. The number of clusters for these two methods was set at two. Two groups of attributes used in this test were computer tasks (CT) and non-computer tasks (NT). The selected attributes for CT included CPU rate, maximum CPU rate, cycles per instruction, and sampled CPU usage. For NT, the attributes were disc I/O time, local disc space usage, and maximum disc I/O time. To detect the best method, we adhered to the criterion that "The clustering algorithm, which divides the workload traces into two groups with an almost equal number of elements, is better to be applied" [7].

The results of using our technique showed that better results could be achieved by using the SOM method to cluster Google workload traces. By implementing this choice, SOM divided the traces into two parts, with a proportion of 61% for CT and 39% for NT. This indicates that SOM was more effective than K-means and density-based methods in segregating the tasks of CT and NT, as illustrated in Table 4.6. This demonstrates that EFection successfully recommended a more efficient clustering method in this study.

**Compare Attribute Selection Against Pattern Extraction Study** In the second experiment, we compared the quality achieved by EFection's suggested attributes against the quality of the attributes used in [70]. In this study, Eva et al. investigated the extraction of characterisation and patterns of Google and Bitbrain workload traces based on CPU and memory

Table 4.6: Comparison between EFection's suggestions and the methods used in a pattern extraction study

| Attributes | No. clusters (Google trace) | No. clusters (Bitbrain) | Reference |
|---|---|---|---|
| CPU | 13 | 15 | Eva et al. [70] |
| CPU + Memory | 18 | 15 | Eva et al. [70] |
| CPU + Memory + Usage + Timestamp | 21 | 17 | EFection suggestions |

utilisation. Two attributes were employed: CPU and Memory Usage. The paper utilised the elbow method to select the optimal predefined number of clusters. The study demonstrated that clustering results are more detailed when using the combination of CPU and Memory Usage compared to using them individually. The criteria employed were able to group the datasets with more detailed characterisation.

By applying the EFection technique, it suggests that even more efficient extraction (clustering) can be achieved by using the combination of CPU, Memory Usage, and Time-stamp instead of the original attributes, as presented in Table 4.6. The EFection suggestion resulted in 21 clusters for Google Trace and 17 for Bitbrain. This is more detailed compared to the original results, which were around 13 to 18 clusters for Google Trace and 15 for Bitbrain. This demonstrates the ability of EFection to detect better attributes than those used in related studies.

### Discussion

The above evaluation experiments have confirmed the applicability of EFection technique and highlighted its potential to significantly improve the quality of outcomes in clustering studies. This advancement will be of particular interest to the field of cloud computing, where extracting critical information through clustering is a vital research focus.

By integrating our technique into their methodology, researchers could be empowered to identify and select the most suitable methods for their clustering studies prior to initiating experimental work. Furthermore, our technique could serve as a valuable tool to pinpoint the most effective clustering implementations, thereby optimising their proposed algorithms and methodologies.

## 4.2 Validation of MICRAST Performance

We conducted the evaluation in this work through two main experiments, as depicted in Figure 4.1. First, we carried out a comparison test to measure the performance of our approach against the LSTM-RNN method in [93]. This case study exemplifies the micro-prediction approach adopted by other related works listed in Table 3.2. We selected this study for comparison since it is similar to our approach in aiming to predict consumption requests using an ANN-based model. Such evaluation is essential to demonstrate the benefit of the proposed approach. Second, we measured the forecasting confidence of MICRAST to show its performance across a range of time steps. This is vital to demonstrate the application scope of our approach. The following subsection presents these two experiments.

### 4.2.1 MICRAST vs LSTM-RNN for Related Work

In this evaluation, we compared the performance of MICRAST with the LSTM-RNN approach. This was conducted for both uni-attribute and multi-attribute forecasting scenarios to ensure comprehensive validation. To measure each approach's performance, we used the $R^2$ and MAPE metrics. As illustrated in Section 2.1.3, we selected these metrics because they provide a clear scale for measuring forecasting accuracy. They assess the degree of alignment between actual and predicted data with a clear and accurate percentage-based value, comparable across different forecasting models. We present the comparison results for each scenario.

Before proceeding to the results, we discuss the experimental configuration. Both forecasting scenarios adhered to the same evaluation setup described in the experimental implementation in Section 3.3.1, using all the selected traces listed in Table 2.12. Accordingly, we first utilised both approaches to predict consumption patterns for the selected attribute, which represents users' usage records (i.e., Requested Number of Processors), as illustrated in Figure 4.1. Second, in the multi-attribute scenario, we repeated the previous steps with one difference: in this case, we trained the forecasting models with the historical records of an additional attribute (i.e., RunTime). Accordingly, we used the history of two attributes from the cloud trace to forecast the value of one particular attribute. We selected these attributes as they reflect major aspects of consumption (demand level and duration).

Table 4.7: Comparison of uni-attribute forecasting results

| Forecasting approach | $R^2$ | MAPE |
|---|---|---|
| LSTM-RNN | 30% | 42.78% |
| MICRAST | 97% | 2.38% |

**Uni-attribute Forecasting Scenario**

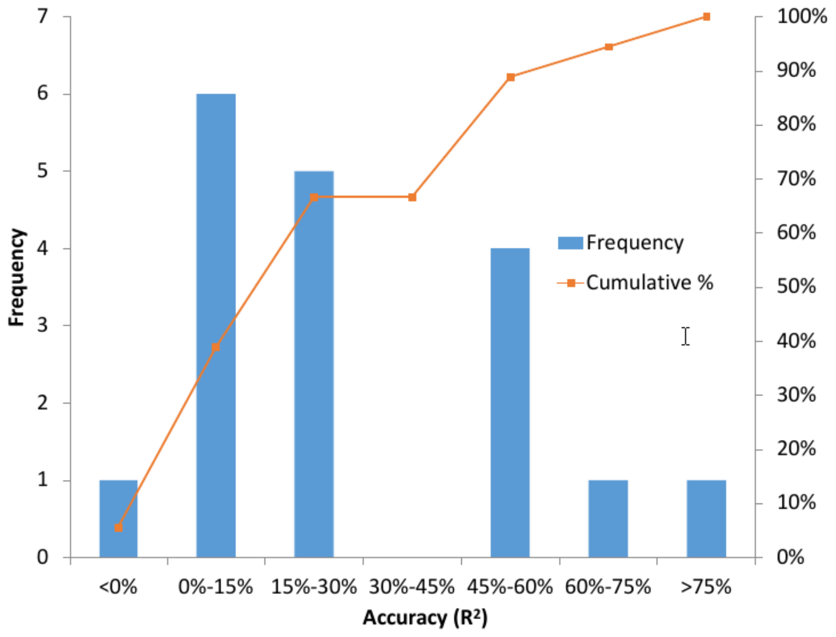
Table 4.7 compares the average $R^2$ and MAPE scores for forecasting all the selected traces by each approach. It demonstrates that our approach achieved better $R^2$ and MAPE by 67% and 40%, respectively. These results show a potentially significant improvement in accuracy when using our approach for uni-attribute forecasting.

For more detailed results, we present the cumulative distribution of $R^2$ scores for both approaches in Figure 4.11. Specifically, the cumulative distribution for the LSTM-RNN approach in Figure 4.7a shows that 16 out of 18 traces have an $R^2$ score below 60%. In contrast, Figure 4.7b demonstrates that MICRAST achieved an $R^2$ score above 90% for 17 of these traces.

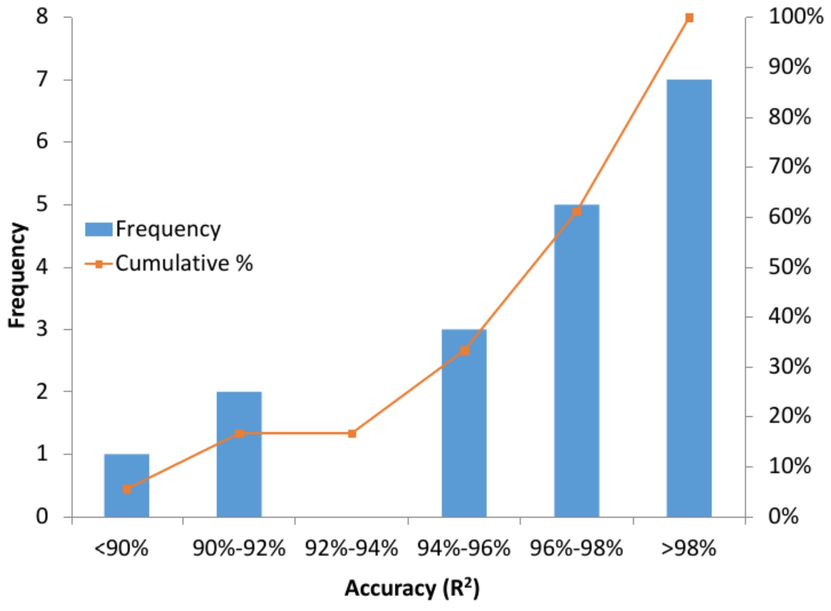
We also demonstrated the MAPE for each trace in Figure 4.8. The related work's approach showed a significantly high MAPE for some traces, specifically around 165% to 209% for forecasting SDSC Par's traces and approximately 124% for ANL-interpad. In contrast, our approach improved all MAPE scores to below 6%, with the majority below 1%, notably for the SDSC Par's and ANL-interpad traces (see Figure 4.8).

The above results are mainly due to the characteristics of cloud traces, as demonstrated in the analysis in Subsection 2.1.2 and Figure 2.2. Some traces exhibited abrupt and unexpected variations with a high standard deviation. Specifically, the standard deviation of the Requested Number of Processors in SDSC Par 1996, 1995, and ANL-Interpad exceeded 10K. Without a suitable extraction process, such characteristics pose a significant challenge for the LSTM-RNN, leading to notably low and unstable performance. In contrast, the performance of our approach indicates that the filtering and clustering processes were highly effective in extracting useful patterns, even from these challenging traces. For example, this is evident in the extracted patterns from the ANL-Interpad trace (shown in Figure 4.9) compared to their raw, pre-extracted form (Figure 2.2). As mentioned earlier, these patterns represent the hidden trends in users' consumption records, which in turn facilitated the learning and prediction process for the RNN model in MICRAST.

Finally, we calculated the relative deviation (RD) for the $R^2$ results of both approaches. We plotted boxplots of these RD distributions in Figure 4.10 to compare the level of consistency of each approach. Accordingly,



(a) LSTM-RNN



(b) MICRAST

Figure 4.7: Comparison of $R^2$ results for uni-attribute forecasting between LSTM-RNN and MICRAST

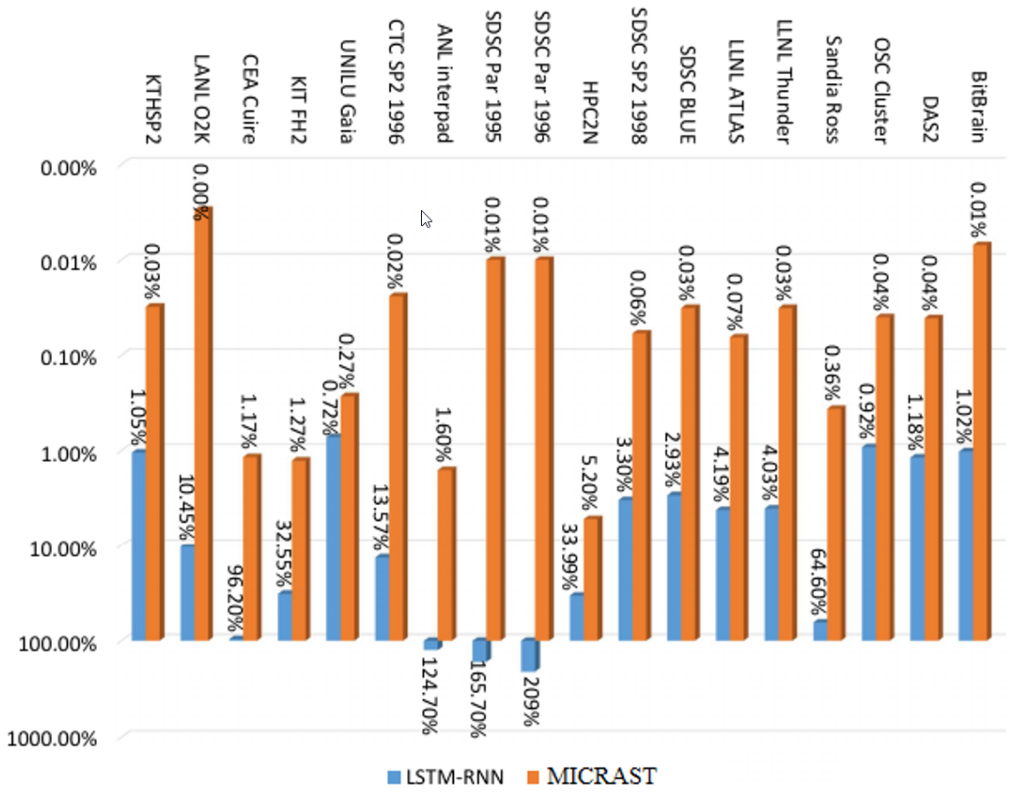


Figure 4.8: Comparison of MAPE results for uni-attribute forecasting

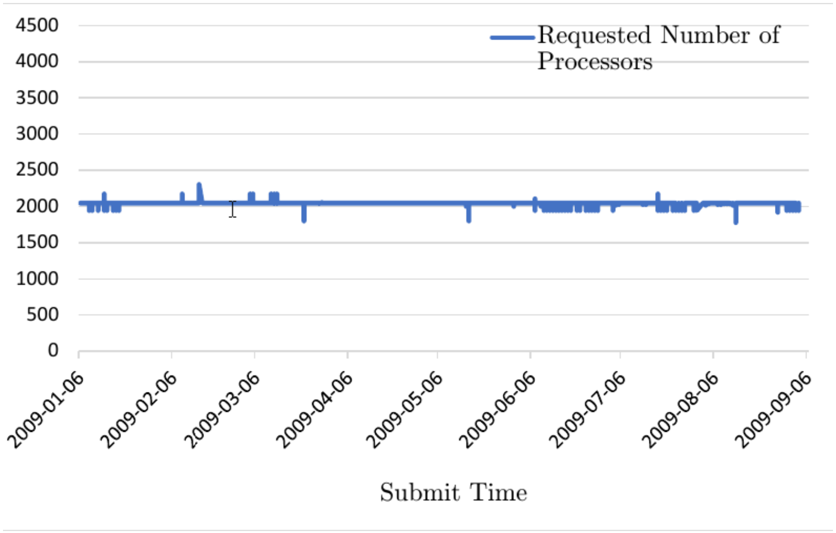


Figure 4.9: Extracted pattern from ANL-Interpad trace attribute

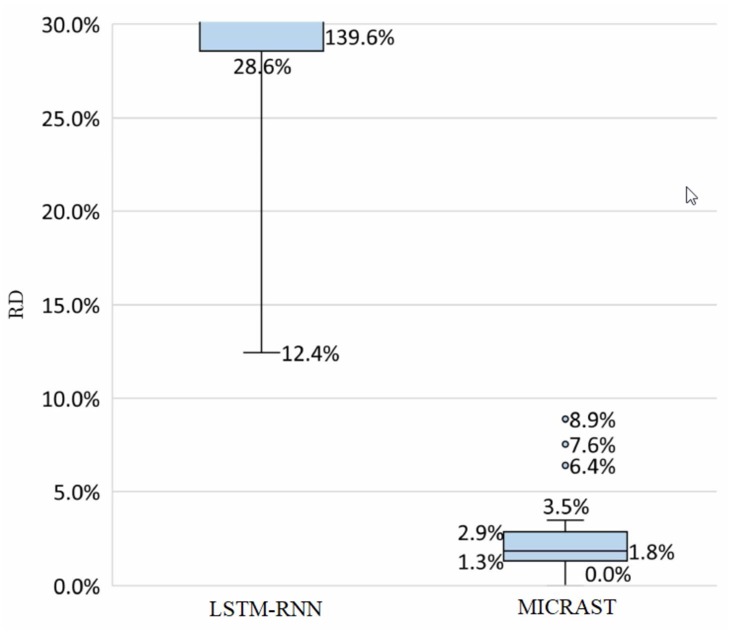


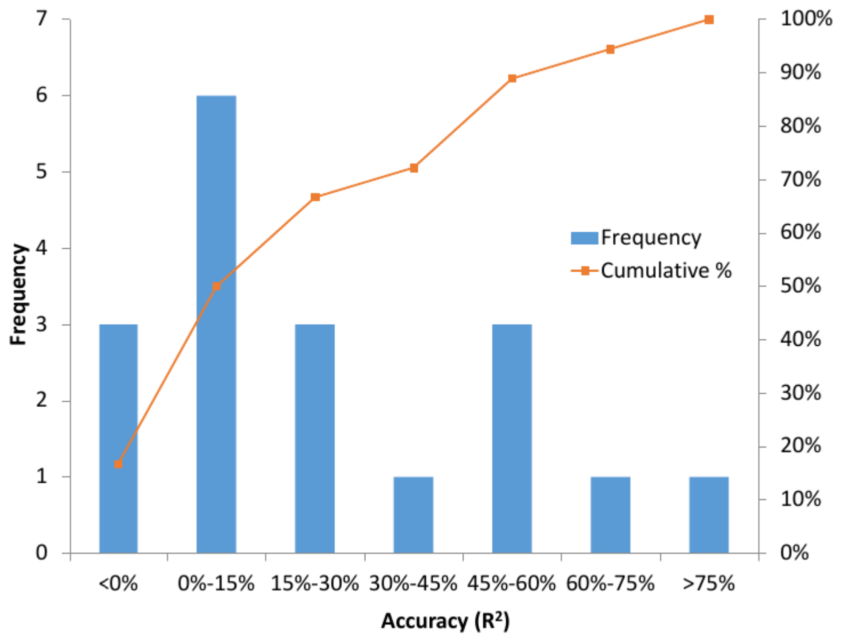
Figure 4.10: Comparison of relative deviation in uni-attribute forecasting

the LSTM-RNN exhibited a wide RD spread of 111 percentage points, while our approach showed a much narrower distribution of just 1.8 percentage points, indicating more centred $R^2$ scores. This narrow distribution, combined with a high average $R^2$ of 97%, demonstrates that our approach delivers more accurate and consistent forecasting in the uni-attribute scenario compared to the related work.

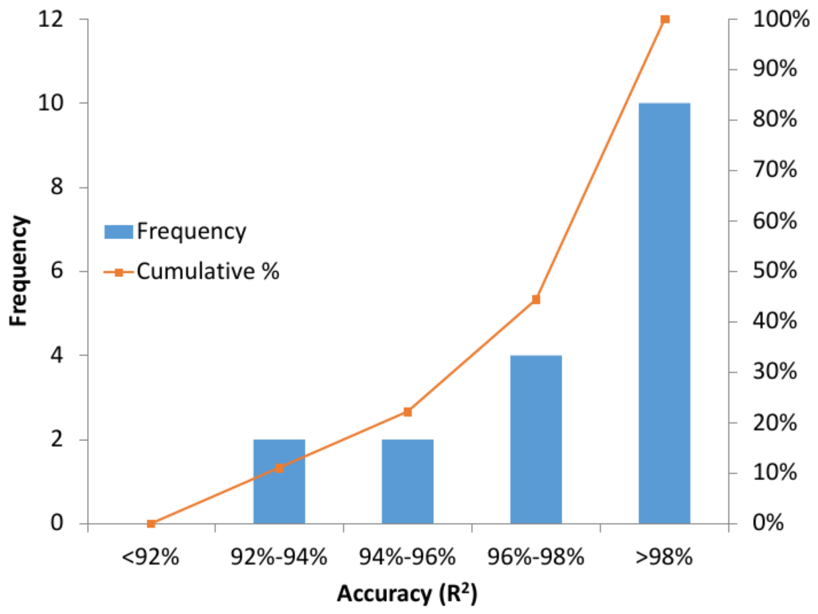
### Multi-attribute Forecasting Scenario

In the second scenario, we observed that the related work exhibited even lower performance than in the previous case. The average $R^2$ dropped down by 3 percent as shown in Table 4.8. The cumulative distribution results in Figure 4.11a show an $R^2$ below 5% for three of the traces—an increase from only one trace in the uni-attribute forecasting scenario. In contrast, our approach maintained its accuracy in the multi-attribute setting, with average $R^2$ around 97% and no traces falling below 90%, as depicted in Figure 4.11b and Table 4.8.

Furthermore, the scores in Figure 4.12 show an even higher MAPE for the LSTM-RNN approach. It recorded approximately 166% to 211% MAPE for SDSC-Par's traces and around 127% for ANL-Interpad. This reflects an increase of about 2 percentage points compared to the uni-attribute forecasting. In contrast, our approach maintained a MAPE below 5.30% across all



(a) LSTM-RNN



(b) MICRAST

Figure 4.11: Comparison of $R^2$ results for multi-attribute forecasting between LSTM-RNN and MICRAST

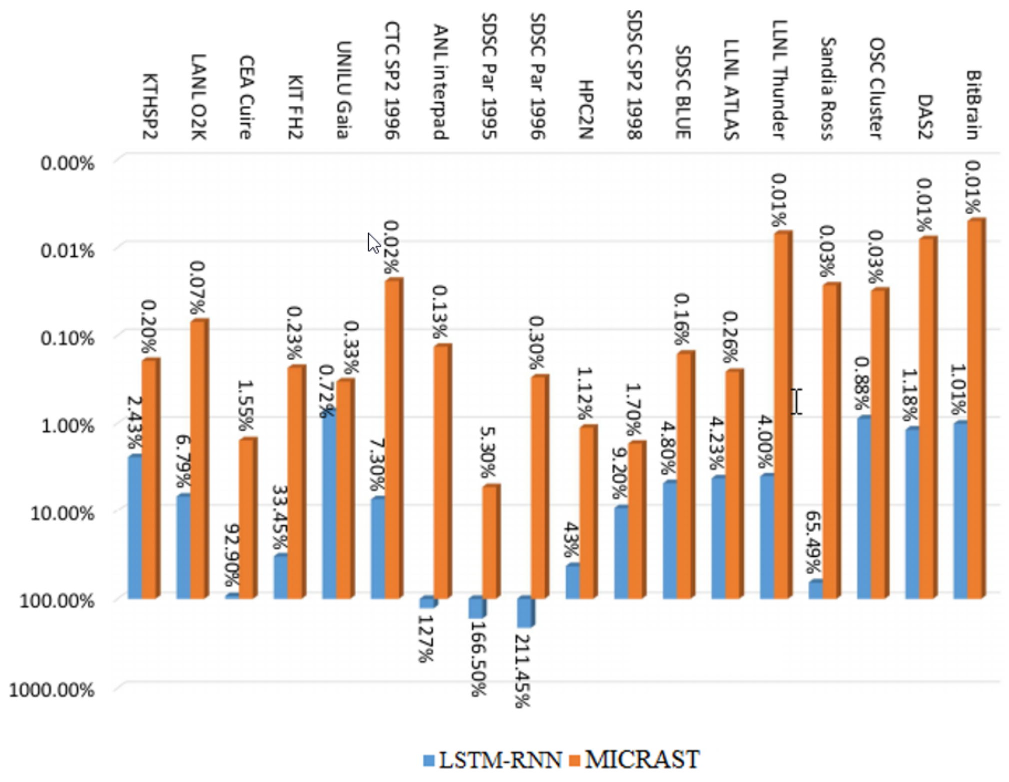


Figure 4.12: Comparison of MAPE results for multi-attribute forecasting

Table 4.8: Comparison of multi-attribute forecasting results

| Forecasting approach | $R^2$ | MAPE |
|---|---|---|
| LSTM-RNN | 27% | 43% |
| MICRAST | 97% | 1% |

traces.

These results are due to challenges caused by the use of multiple attributes with sudden change characteristics. Such characteristics make it difficult for the LSTM-RNN approach to capture possible correlations between these attributes, as they fail to provide meaningful patterns. In contrast, the extraction phase in MICRAST enables the uncovering of detailed attribute patterns through clustering, making it easier for the prediction model (i.e., the RNN model) to identify potential correlations.

The boxplots for the relative deviation distribution of both approaches in Figure 4.13 show that LSTM-RNN failed to adapt to this type of forecasting. It recorded a relative deviation spread of 212 percentage points, which is approximately 100 percentage points higher than in the uni-attribute fore-

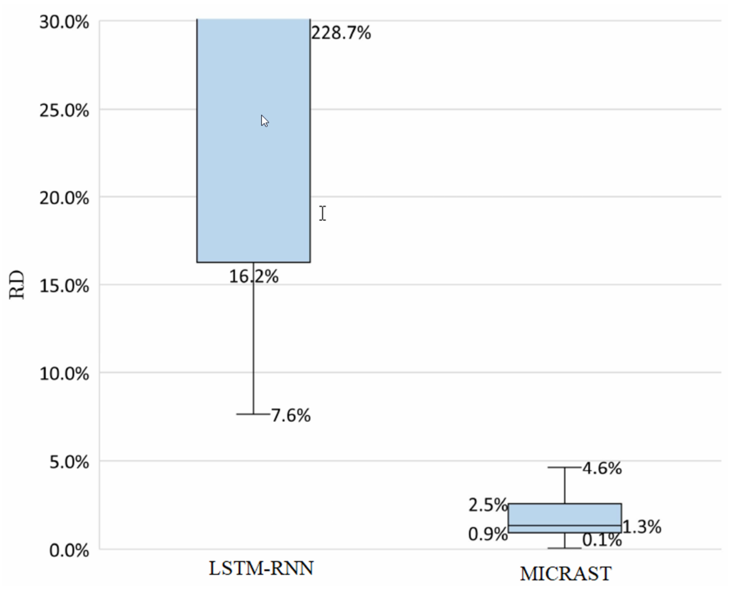


Figure 4.13: Comparison of relative deviation in multi-attribute forecasting

casting scenario. In contrast, our approach maintains consistency in multi-attribute forecasting, with the relative deviation spread of only 1.6 percentage points.

## 4.2.2 Confidence range for MICRAST

In this experiment, we measured forecasting confidence by demonstrating the change in $R^2$ values for our approach as we extended the range of the forecast. We varied the range from 0.05% to 20% of each trace's training data (e.g., if the training data was 1 hour long, we made forecasts from 18 seconds to 9 minutes into the future). We have chosen this range because our observations showed that within this range there are significant chances of consumption pattern changes for each trace. Therefore, evaluating across the complete range demonstrates our ability to cope with forecasting even these changes.

We applied the same experimental configurations as in the previous evaluation. Similarly, we conducted uni-attribute forecasting of users' consumption patterns of Requested Number of Processors for all the selected traces in Table 2.12. Finally, we calculated the median of these traces' $R^2$ for each step. Ultimately, the ($R^2$-median, $R^2$) over a particular forecasting range gives our MICRAST confidence.

The results in Figure 4.14 show that our approach forecasted the majority of the traces with $R^2$ distributed within a range of 5 percentage points around the median of 98% $R^2$. This range expanded to 19 percentage points around the median of 93% $R^2$ when reaching 20% of the steps in the training data. This expansion is mainly noticed in the traces of DAS2 and ANL-Interpad. As mentioned previously, these traces exhibit a significant characteristic of sudden changes in their consumption patterns, as illustrated for the ANL-Interpad trace in Figure 2.2. This characteristic raises more challenges for the RNN model when the time step increases, even after the extraction process, affecting the prediction quality over time. Nevertheless, Figure 4.14 shows that our approach can maintain the high $R^2$ median around 95% to 98% for the majority of the traces, while it drops by only 5 percentage points (to 93%) when reaching the full 20% of the rows from the training trace. This demonstrates that predictions up to 4% of the trace can be relied on for all traces, while for most traces we can reliably predict even 20% into the future of the training data.

## 4.3 Findings

In this chapter, we proposed an approach, MICRAST, for forecasting users' patterns in a cloud environment based on their consumption patterns. Our approach conducts this by extracting these patterns from the input traces through filtering and clustering processes. Then, it standardises them through time alignment, linear interpolation, and normalisation. Finally, our approach passes the standardised patterns to an RNN model for forecasting, which we selected through a preliminary experiment. When comparing our work with prior art, we demonstrated that such extraction and pre-processing in MICRAST enables it to provide more efficient predictions for traces that exhibit characteristics of abrupt changes.

We evaluated the MICRAST approach through the following experiments. First, we compared our approach against those used in related works (i.e., LSTM-RNN) to demonstrate its superiority. Our approach showed the ability to conduct both univariate and multivariate forecasting with an accuracy of 98%, surpassing the LSTM-RNN approach by around 70 percentage points. Second, we measured the confidence range of our approach by observing how accuracy changed when we increased how far ahead the forecasting needed to go. The results show that MICRAST was able to forecast users' patterns with a confidence level between 95% and 98% when forecasting for a duration of 20% of the training data.

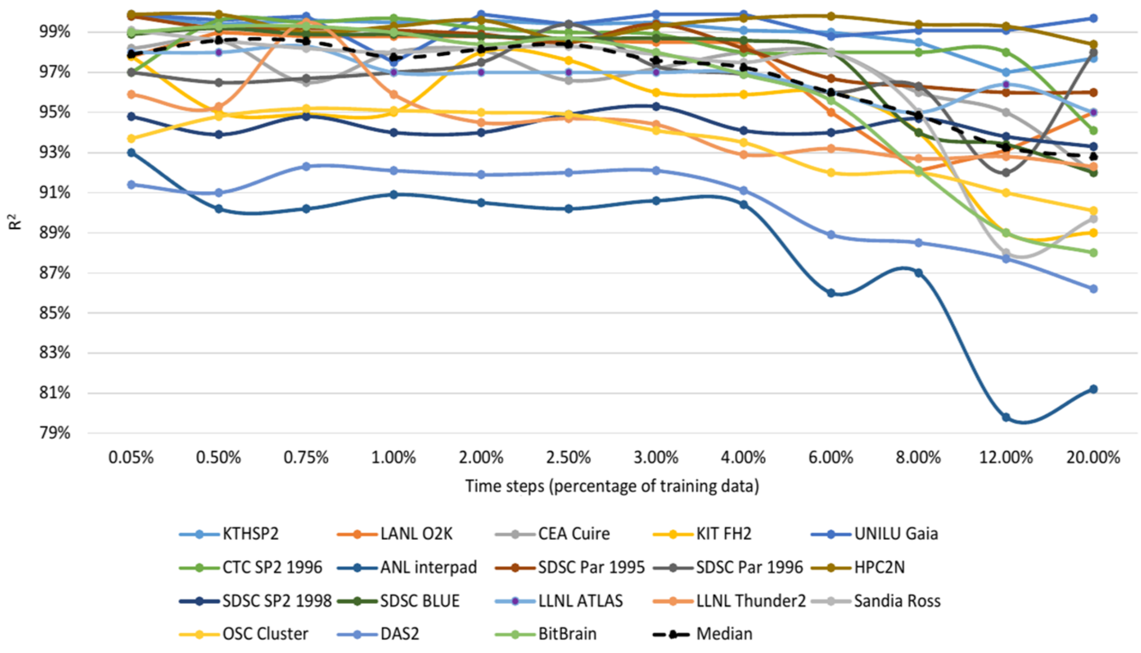


Figure 4.14: Confidence range of the MICRAST approach over time

## 4.4 Summary

In this chapter, we covered the evaluation and results for our extraction and forecasting tools. First, we validated SeQual by testing its ability to perform unsupervised selection of attributes in cloud traces. This involved comparing it against feature selection methods such as RF, PCF, and LS. SeQual outperformed related methods by around 8% to 28% in accuracy and showed promising performance for unsupervised traces as well. Then, we evaluated our second extraction tool, the EFection technique, through three main aspects: accuracy, applicability, and superiority. Our technique demonstrated the ability to pre-detect the attribute combinations and methods likely to yield the highest clustering quality. It matched the optimal choice in about 83% of the input samples and surpassed the performance of the most recent attempt by 11 percentage points, with better stability. Both extraction tools achieve this performance without requiring supervisory inputs or parameters such as the number of clusters or expected classifications. This makes our tools beneficial for datasets that do not provide such information.

Finally, we validated our forecasting tool, MICRAST, by comparing it against related work methodologies (i.e., LSTM-RNN). Then, we measured the confidence range of MICRAST by observing how accuracy changed when increasing the size of the forecasting steps. The results showed our approach's ability to achieve micro-prediction of consumption patterns in the traces with an accuracy of 98%, surpassing the LSTM-RNN by around 70 percentage points. It also showed a confidence level between 95% and 98% for a range of time steps equivalent to 20% of the training data.

To wrap up, the evaluation and results in this chapter underscore the effectiveness of our extraction and forecasting tools. These findings highlight the robustness and applicability of our methods, setting a solid foundation for future research and development. In the next chapter, we will highlight these contributions and provide conclusions on their implications for future work.

# Chapter 5

# Conclusion

## 5.1 Summary

In an attempt to obtain better cloud resource management, many studies have been used to extract and predict vital information from cloud records through clustering and forecasting tools. Such studies can assist greatly in steering users towards more aware usage if they target the patterns in these records at a detailed level.

Therefore, in Chapter 2, we initially conducted a thorough investigation. This investigation showed the necessity of providing selection tools for clustering attributes and methods and developing an approach to forecasting that enables better detailed extraction and prediction of cloud patterns.

As a result, this thesis presented in Chapter 3 an analysis tools that provide the aforementioned abilities. For single-attribute selection, we proposed the SeQual method that ranks the candidate attributes for clustering by exploiting the ability of the silhouette coefficient metric. While, for multi-attributes and clustering method pre-detection, we developed EFection, which accomplishes this by using a combination of internal validation metrics DB and the Coefficient of Determination. Whereas, for efficient prediction, we proposed the MICRAST which captures and predicts the detailed patterns from cloud traces. Our approach accomplishes this by wrapping up our previous extraction tool with phases of uniforming and time alignment.

The evaluation results in Chapter 4 demonstrated the performance of our extraction and forecasting tools. They showed that SeQual can compete with the supervised selection methods and perform better than unsupervised ones by around 8% to 28%. The results also supported the ability of the EFection technique to offer automated detection with high accuracy, around 83%, surpassing prior art by 15%. On the other hand, the assessment of

MICRAST demonstrated its ability to forecast detailed patterns with a level of accuracy between 95% and 98%, outdoing related works by approximately 70%.

## 5.2 Contribution to Science

The new scientific findings of this dissertation are presented as follows:

- *My proposed method of clustering attribute selection, SeQual, performs more accurately without requiring supervisory inputs. It asks only for the data as input, making it more applicable to cloud traces that do not provide much information. To enable such selection, I introduced sample clustering to SeQual. This applies clustering to a sample of the input attributes across a range of expected cluster numbers k. Then, I employed the silhouette coefficient in the algorithm to form a scale of quality for each value of k across these attributes. Finally, SeQual examines this scale for the highest average silhouette score and identifies the pattern of peaks and troughs to rank each attribute. The evaluation results support that SeQual delivers higher accuracy compared to related methods by around 8% to 28%.*
  **Related Publications: [P1], [P3], [P5]**
  **Related Sections: Section 3.2.1 and Section 4.1**

- *My technique of pre-detecting multi-attribute combinations and clustering methods, EFection, operates without merging these attributes or involving manual intervention. By doing so, it preserves the originality of the information to be extracted and ensures its reliability for repeated tasks. To achieve such pre-detection, I utilised both the Davies–Bouldin index and the $R^2$ metric to analyse the quality of clustering samples for different attribute–method combinations. The validation results demonstrate that my EFection outperforms related works by approximately 15%.*
  **Related Publications: [P2], [P4]**
  **Related Sections: Section 3.2.2 and Section 4.1**

- *My new approach to micro-prediction, MICRAST, performs such forecasting for cloud resource consumption by training the models on pre-processed, detailed patterns extracted from cloud traces. It produces a trained network for each of the extracted patterns, which is then used to forecast the input trace. To support such prediction, I integrated both*

*SeQual and EFection during the extraction phase, along with uniforming and time alignment for pre-processing. The validation of MICRAST shows that it achieves a confidence level between 95% and 98%, providing around 70% higher accuracy compared to existing macro-prediction approaches.*
**Related Publication: [P6]**
**Related Sections: Section 3.3 and Section 4.2**

## 5.3 Recommendations and Future Work

For future work, we have identified the following directions for our analysis tools. We plan to investigate whether the integration of other internal validation metrics, such as the Dunn Index, can enhance the accuracy of detection within our extraction tools, SeQual and EFection. Another potential application of the EFection technique is anomaly detection in cloud computing, which is currently beyond the scope of this thesis. This process involves clustering cloud workloads based on 3 to 7 key dimensions, such as resource usage, execution time, and job type. By analysing jobs, the technique can identify clusters representing normal behaviour and flag any data points that significantly deviate from these patterns as anomalies. The investigation of MICRAST should also be extended by implementing it in cloud simulators such as CloudSim and DISSECT-CF. This implementation would test its ability to influence user behaviour in real-world scenarios and under more challenging conditions.

Furthermore, we plan to test our tools on more diverse types of datasets beyond scientific cloud traces (e.g., web applications, serverless cloud functions, IoT systems, or platform-specific services like Azure). This will support the applicability of these tools for more general use. It is expected that our tools will be applicable to these traces that exhibit general characteristics, shown in Table 2.3. These characteristics are similar to those of the traces used in this thesis for validation. However, using such data may raise the possibility of prediction hallucination, as discussed in Section 2.3. To mitigate this, it is recommended that future systems incorporate a human-in-the-loop mechanism. In this approach, users or domain experts would have the ability to review forecast outputs, provide contextual input, or override model predictions when necessary. Introducing this form of human oversight could improve the reliability and contextual relevance of the predictions, while also supporting ethical alignment. This recommendation aims to reduce over-reliance on automated decisions and promote transparency, trust, and accountability in data-driven forecasting systems.

It is also recommended to use these tools in combination as a framework to provide deeper behavioural insights in resource-related studies. This framework would enable more human-centred analysis in such works. By offering the extraction and forecasting of users' patterns (at a more individual level), the framework allows these studies to achieve improved resource management.

## 5.4 Publication Related to This Dissertation

**[P1]** Ali, Shallaw Mohammed, and Gabor Kecskemeti. ”SeQual: an unsupervised feature selection method for cloud workload traces.” The Journal of Supercomputing 79.13 (2023): 15079-15097. **Scoups indexed[Q2]**.

**[P2]** Ali, Shallaw Mohammed, and Gabor Kecskemeti. ”EFection: Effectiveness Detection Technique for Clustering Cloud Workload Traces.” International Journal of Computational Intelligence Systems 17.1 (2024): 202. **Scopus indexed[Q2].**
**[P3]** Ali, Shallaw Mohammed, and Gabor Kecskemeti. ”Clustering datasets in cloud computing environment for user identification.” 2022 30th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP). IEEE, 2022. **Scopus indexed.**

**[P4]** Ali, Shallaw Mohammed, Gabor Kecskemeti. ”Cloud user Behavior prediction for resources usage awareness” In: Molnár, Dániel; Molnár, Dóra (eds.) XXIV. Tavaszi Szél Konferencia 2021: Absztrakt kötet Bp, Hungary : Association of Hungarian PHD and DLA Students (2021) 667 p.p. 401.

**[P5]** Ali, Shallaw Mohammed, Gábor Kecskeméti. ”The Prediction and Analysis of Cloud User Behavior for Resources Usage Awareness” In: Ivanyi, Peter (eds.) Abstract book for the 17th MIKLOS IVANYI INTERNATIONAL PHD and DLA SYMPOSIUM : ARCHITECTURAL, ENGINEERING AND INFORMATION SCIENCES Pécs, Hungary : Pollack Press (2021) 227 p. p. 106.

**[P6]** Ali, Shallaw Mohammed, Gábor Kecskeméti. ”MICRAST: Microforecasting approach for cloud user consumption pattern based on RNN” International Journal of Advanced Computer Science and Applications,, 2025. **Scopus indexed[Q3].**

# Bibliography

[1] K. Mukdasai, Z. Sabir, M. A. Z. Raja, R. Sadat, M. R. Ali, and P. Singkibud. A numerical simulation of the fractional order leptospirosis model using the supervised neural network. *Alexandria Engineering Journal*, 61(12):12431–12441, 2022.

[2] M. Sadaf, S. Arshed, G. Akram, M. R. Ali, and I. Bano. Analytical investigation and graphical simulations for the solitary wave behavior of chaffee–infante equation. *Results in Physics*, 54:107097, 2023.

[3] Z. Zhou, J. H. Abawajy, and F. Li. *Analysis of energy consumption model in cloud computing environments*. 2019.

[4] S. Ostermann, G. Kecskemeti, and R. Prodan. Fostering energy-awareness in scientific cloud users. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 149–154. IEEE, 2014.

[5] S. Arora and I. Chana. A survey of clustering techniques for big data analysis. In *2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence)*, pages 59–65. IEEE, 2014.

[6] M. Kureljusic and E. Karger. Forecasting in financial accounting with artificial intelligence–a systematic literature review and future research agenda. *Journal of Applied Accounting Research*, 2023. ahead-of-print.

[7] S. A. Yousif and A. Al-Dulaimy. Clustering cloud workload traces to improve the performance of cloud data centers. In *Proceedings of the World Congress on Engineering*, volume 1, pages 7–10, 2017.

[8] Nagma Khattar, Jagpreet Sidhu, and Jaiteg Singh. Toward energy-efficient cloud computing: a survey of dynamic power management and heuristics-based optimization techniques. *The Journal of Supercomputing*, 75:4750–4810, 2019.

[9] Hong Huang, Yu Wang, Yue Cai, and Hong Wang. A novel approach for energy consumption management in cloud centers based on adaptive fuzzy neural systems. *Cluster Computing*, 27:14515–14538, 2024.

[10] F. Chen, J. Grundy, J.-G. Schneider, Y. Yang, and Q. He. Automated analysis of performance and energy consumption for cloud applications. In *Proceedings of the 5th ACM/SPEC international conference on Performance engineering*, pages 39–50, 2014.

[11] J.W. Park and E. Kim. Runtime prediction of parallel applications with workload-aware clustering. *The Journal of Supercomputing*, 73(11):4635–4651, 2017.

[12] H.-D. Meng, J.-H. Ma, and G.-D. Xu. Experimental research on impacts of dimensionality on clustering algorithms. In *2010 International Conference on Computational Intelligence and Software Engineering*, pages 1–4. IEEE, 2010.

[13] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues. Clustering algorithms: A comparative approach. *PloS one*, 14(1):e0210236, 2019.

[14] S. Barak and T. Mokfi. Evaluation and selection of clustering methods using a hybrid group mcdm. *Expert Systems with Applications*, 138:1027–1035, 2019.

[15] S. Chormunge and S. Jena. Correlation based feature selection with clustering for high dimensional data. *Journal of Electrical Systems and Information Technology*, 5(3):542–549, 2018.

[16] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[17] S. Solorio-Fernandez, J. A. Carrasco-Ochoa, and J. F. Martinez-Trinidad. A review of unsupervised feature selection methods. *Artificial Intelligence Review*, 53(2):907–948, 2020.

[18] E. Hancer, B. Xue, and M. Zhang. A survey on feature selection approaches for clustering. *Artificial Intelligence Review*, 53(6):4519–4545, 2020.

[19] Y. S. Patel and J. Bedi. Mag-d: A multivariate attention network based approach for cloud workload forecasting. *Future Generation Computer Systems*, 142:376–392, 2023.

[20] Y. Lu, J. Panneerselvam, L. Liu, Y. Wu, et al. Rvlbpnn: A workload forecasting model for smart cloud computing. *Scientific Programming*, 2016, 2016.

[21] B. Feng, Z. Ding, and C. Jiang. Fast: A forecasting model with adaptive sliding window and time locality integration for dynamic cloud workloads. *IEEE Transactions on Services Computing*, 16(2):1184–1197, 2022.

[22] Y. Sfakianakis, E. Kanellou, M. Marazakis, and A. Bilas. Trace-based workload generation and execution. In *Euro-Par 2021: Parallel Processing: 27th International Conference on Parallel and Distributed Computing*, pages 37–54, Lisbon, Portugal, 2021. Springer. Proceedings 27, September 1–3, 2021.

[23] S. Anoep, C. Dumitrescu, D. Epema, A. Iosup, M. Jan, H. Li, and L. Wolters. The gird workloads archive, 2023. Accessed: 2024-11-27.

[24] D. Feitelson. The parallel workloads archive, 2021.

[25] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. Epema. The grid workloads archive. *Future Generation Computer Systems: the international journal of grid computing: theory, methods and applications*, 24:672–686, 2008.

[26] L. Rokach and O. Maimon. *Clustering methods*. 2005.

[27] A. B. Ayed, M. B. Halima, and A. M. Alimi. Survey on clustering methods: Towards fuzzy clustering for big data. In *2014 6th International conference of soft computing and pattern recognition (SoCPaR)*, pages 331–336. IEEE, 2014.

[28] M. Halkidi. *Hierarchical clustering*, pages 588–594. Springer, 2017.

[29] H. S. B. Janmenjoy Nayak and Bighnaraj Naik. Fuzzy c-means (fcm) clustering algorithm: A decade review from 2000 to 2014. In *Computational Intelligence in Data Mining—Volume 2*, pages 131–138. Springer, 2015.

[30] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of intelligent information systems*, 17(2-3):107–145, 2001.

[31] N. Tomasev and M. Radovanovic. Clustering evaluation in high-dimensional data. In *Unsupervised learning algorithms*, pages 71–107. Springer, 2016.

[32] C. Liu, R. W. White, and S. Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 379–386. ACM, 2010.

[33] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[34] Michael Steinbach George Karypis, Vipin Kumar, and Michael Steinbach. A comparison of document clustering techniques. In *TextMining Workshop at KDD2000 (May 2000)*, pages 428–439, 2000.

[35] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[36] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation, 2020. arXiv preprint arXiv:2010.16061.

[37] J. M. Santos and M. Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *Artificial Neural Networks–ICANN 2009: 19th International Conference, Limassol, Cyprus, September 14-17, 2009, Proceedings, Part II 19*, pages 175–184. Springer, 2009.

[38] J. Cai, J. Luo, S. Wang, and S. Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018.

[39] J. Miao and L. Niu. A survey on feature selection. *Procedia Computer Science*, 91:919–926, 2016.

[40] K. Kira, L. A. Rendell, et al. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, volume 2, pages 129–134, 1992.

[41] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

[42] N. El Aboudi and L. Benhlima. Review on wrapper feature selection approaches. In *2016 International Conference on Engineering & MIS (ICEMIS)*, pages 1–5. IEEE, 2016.

[43] A. Mackiewicz and W. Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.

[44] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008.

[45] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[46] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56:9–33, 2004.

[47] C. Chatfield. *Time-series forecasting*. CRC Press, 2000.

[48] I. Svetunkov and F. Petropoulos. Old dog, new tricks: a modelling view of simple moving averages. *International Journal of Production Research*, 56(18):6034–6047, 2018.

[49] Investopedia. Simple moving average (sma) definition, Dec 2021. Accessed on 2024-01-13.

[50] R. J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice (3rd ed)*. 2023. Accessed on 2024-01-15.

[51] R. H. Shumway and D. S. Stoffer. *ARIMA models*. Springer, 2017.

[52] K. Holden. Vector auto regression modeling and forecasting. *Journal of Forecasting*, 14(3):159–166, 1995.

[53] L. Fausett. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall International Editions, 1994.

[54] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[55] Baeldung. Prevent the vanishing gradient problem with lstm, 2024. Accessed on 2024-01-15.

[56] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 2014.

[57] R. Nau. Statistical forecasting: Notes on regression and time series analysis, 2020. Fuqua School of Business, Duke University.

[58] WallStreetMojo. Unit root tests - definition, types, examples, and advantages, 2021. Accessed on 2024-03-03.

[59] D. A. Dickey and W. A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366a):427–431, 1979.

[60] Q.-X. Zheng, Y.-L. Wang, P. Lu, S.-L. Liu, Y. Zhou, and J.-G. Zheng. Automatic time-shift alignment method for chromatographic data analysis. *Scientific Reports*, 7:3907, 2017.

[61] Mathful. Linear interpolation: Definition, formula, & example, 2024. Accessed on 2024-05-18.

[62] A. I. Magazine. A guide to different evaluation metrics for time series forecasting models, 2021. Accessed on 2024-03-03.

[63] S. Glen. Absolute error: Definition, formula, examples, 2020. Accessed on 2024-01-15.

[64] S. Allwright. How to interpret mape (simply explained), 2022. Accessed on 2024-01-15.

[65] V. Profillidis and G. Botzoris. *Chapter 5—Statistical methods for transport demand modeling.* 2019.

[66] Y. Yu, V. Jindal, I.-L. Yen, and F. Bastani. Integrating clustering and learning for improved workload prediction in the cloud. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 876–879. IEEE, 2016.

[67] M. Ghobaei-Arani. A workload clustering based resource provisioning mechanism using biogeography based optimization technique in the cloud based systems. *Soft Computing*, 25(5):3813–3830, 2021.

[68] J. Gao, H. Wang, and H. Shen. Machine learning based workload prediction in cloud computing. In *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE, 2020.

[69] A. Shahidinejad, M. Ghobaei-Arani, and M. Masdari. Resource provisioning using workload clustering in cloud computing environment: a hybrid approach. *Cluster Computing*, 24(1):319–342, 2021.

[70] E. Patel and D. S. Kushwaha. Clustering cloud workloads: K-means vs gaussian mixture model. *Procedia Computer Science*, 171:158–167, 2020.

[71] D. Mosoti, V. O. Omwenga, and P. Ogao. Mff: Performance interference-aware vm placement algorithm for reducing energy consumption in data centers. *Open Journal for Information Technology*, 3(1):1, 2020.

[72] S. Ismaeel, A. Miri, and A. Al-Khazraji. Energy-consumption clustering in cloud data centre. In *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, pages 1–6. IEEE, 2016.

[73] A. Li, Y. Meng, and P. Wang. Similarity-based three-way clustering by using dimensionality reduction. *Mathematics*, 12(13):1951, 2024.

[74] Y. Kang, E. Liu, K. Zou, X. Wang, and H. Zhang. Sparse clustering algorithm based on multi-domain dimensionality reduction autoencoder. *Mathematics*, 12(10):1526, 2024.

[75] C. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas. Randomized dimensionality reduction for k-means clustering. *IEEE Transactions on Information Theory*, 61(2):1045–1062, 2014.

[76] R. W. Sembiring, J. M. Zain, and A. Embong. Dimension reduction of health data clustering. *arXiv preprint arXiv:1110.3569*, pages 1041–1050, 2011.

[77] S. Prabhakaran. Principal component analysis – how pca algorithms works, the concept, math and implementation, 2021. Accessed on 2024-01-30.

[78] M. Daraghmeh, A. Agarwal, and Y. Jararweh. An ensemble clustering approach for modeling hidden categorization perspectives for cloud workloads. *Cluster Computing*, pages 1–25, 2023.

[79] H. Jia and Y. Weng. Unsupervised feature selection via adaptive feature clustering for high-dimensional data. In *2022 5th International Conference on Data Science and Information Technology (DSIT)*, pages 1–6. IEEE, 2022.

[80] S. Kumar, N. Muthiyan, S. Gupta, A. Dileep, and A. Nigam. Association learning based hybrid model for cloud workload prediction. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[81] M. S. Jassas and Q. H. Mahmoud. Evaluation of a failure prediction model for large scale cloud applications. In *Canadian Conference on Artificial Intelligence*, pages 321–327. Springer, 2020.

[82] H. Li, D. Groep, and L. Wolters. An evaluation of learning and heuristic techniques for application run time predictions. In *Proceedings of 11th Annual Conference of the Advance School for Computing and Imaging (ASCI)*, Netherlands, 2005. Citeseer.

[83] P. Bhagtya, S. Raghavan, and K. Chandraseakran. Workload classification in multi-vm cloud environment using deep neural network model. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 79–82, 2021.

[84] D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2:165–193, 2015.

[85] D. Hsu. Comparison of integrated clustering methods for accurate and stable prediction of building energy consumption data. *Applied Energy*, 160:153–163, 2015.

[86] M. A. Sanchez, O. Castillo, J. R. Castro, and P. Melin. Fuzzy granular gravitational clustering algorithm for multivariate data. *Information Sciences*, 279:498–511, 2014.

[87] S. Crase and S. N. Thennadil. An analysis framework for clustering algorithm selection with applications to spectroscopy. *PLOS ONE*, 17(3):e0266369, 2022.

[88] W. Wu, Z. Xu, G. Kou, and Y. Shi. Decision-making support for the evaluation of clustering algorithms based on mcdm. *Complexity*, 2020(1):9602526, 2020.

[89] G. Kou, Y. Peng, and G. Wang. Evaluation of clustering algorithms for financial risk analysis using mcdm methods. *Information Sciences*, 275:1–12, 2014.

[90] Y. Lu, L. Liu, J. Panneerselvam, X. Zhai, X. Sun, and N. Antonopoulos. Latency-based analytic approach to forecast cloud workload trend for

sustainable datacenters. *IEEE Transactions on Sustainable Computing*, 5(3):308–318, 2019.

[91] A. I. Maiyza, N. O. Korany, K. Banawan, H. A. Hassan, and W. M. Sheta. Vtgan: Hybrid generative adversarial networks for cloud workload prediction. *Journal of Cloud Computing*, 12(1):97, 2023.

[92] S. Arbat, V. K. Jayakumar, J. Lee, W. Wang, and I. K. Kim. Wasserstein adversarial transformer for cloud workload prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 12433–12439, 2022.

[93] J. Kumar, R. Goomer, and A. K. Singh. Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters. In *Procedia Computer Science*, volume 125, pages 676–682, 2018.

[94] J. Kumar, A. K. Singh, and R. Buyya. Self directed learning based workload forecasting model for cloud resource management. *Information Sciences*, 543:345–366, 2021.

[95] J. Panneerselvam, L. Liu, and N. Antonopoulos. Inot-repcon: Forecasting user behavioural trend in large-scale cloud environments. *Future Generation Computer Systems*, 80:322–341, 2018.

[96] P. Nehra and N. Kesswani. A workload prediction model for reducing service level agreement violations in cloud data centers. *Decision Analytics Journal*, 11:100463, 2024.

[97] Zihang Qiu, Chaojie Li, Zhongyang Wang, Renyou Xie, Borui Zhang, Huadong Mo, Guo Chen, and Zhaoyang Dong. Ef-llm: Energy forecasting llm with ai-assisted automation, enhanced sparse prediction, hallucination detection. *arXiv preprint arXiv:2411.00852*, 2024.

[98] John R Taylor. An introduction to error analysis: The study of uncertainties in physical measurements, 1997.

[99] Shallaw Mohammed Ali and Gabor Kecskemeti. Sequal: an unsupervised feature selection method for cloud workload traces. *The Journal of Supercomputing*, 79(13):15079–15097, 2023.

[100] Shallaw Mohammed Ali and Gabor Kecskemeti. Efection: Effectiveness detection technique for clustering cloud workload traces. *International Journal of Computational Intelligence Systems*, 17(1):202, 2024.

[101] T. Szandala. Review and comparison of commonly used activation functions for deep neural networks. *Bio-inspired neurocomputing*, pages 203–224, 2021.

[102] Shallaw Mohammed Ali and Gabor Kecskemeti. Clustering datasets in cloud computing environment for user identification. In *2022 30th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 165–171. IEEE, 2022.

[103] Mustafa Daraghmeh, Anjali Agarwal, Ricardo Manzano, and Marzia Zaman. Time series forecasting using facebook prophet for cloud resource management. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2021.

[104] Soukaina Ouhame, Youssef Hadi, and Arif Ullah. An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model. *Neural Computing and Applications*, 33(16):10043–10055, 2021.

[105] Minxian Xu, Chenghao Song, Huaming Wu, Sukhpal Singh Gill, Kejiang Ye, and Chengzhong Xu. esdnn: deep neural network based multivariate workload prediction in cloud computing environments. *ACM Transactions on Internet Technology (TOIT)*, 22(3):1–24, 2022.