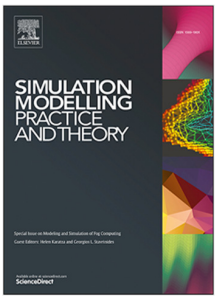
Journal Pre-proof



Optimizing serverless computing: A comparative analysis of multi-output regression models for predictive function invocations

Mustafa Daraghmeh, Anjali Agarwal, Yaser Jararweh

PII: S1569-190X(24)00039-X
DOI: https://doi.org/10.1016/j.simpat.2024.102925
Reference: SIMPAT 102925

To appear in: *Simulation Modelling Practice and Theory*

Please cite this article as: M. Daraghmeh, A. Agarwal and Y. Jararweh, Optimizing serverless computing: A comparative analysis of multi-output regression models for predictive function invocations, *Simulation Modelling Practice and Theory* (2024), doi: https://doi.org/10.1016/j.simpat.2024.102925.

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Journal Pre-proof

Highlights

Optimizing Serverless Computing: A Comparative Analysis of Multi-Output Regression Models for Predictive Function Invocations

Mustafa Daraghmeh, Anjali Agarwal, Yaser Jararweh

- Developed a comprehensive multi-level predictive analysis using multi-output regression models to simulate and predict function invocation patterns in serverless computing, enhancing operational predictability and efficiency.
- Conducted a detailed exploration of the windowing technique, assessing the impact of different configurations on predictive model performance, emphasizing the importance of temporal granularity in workload modeling.
- Investigated the effects of dimensionality reduction using Principal Component Analysis (PCA), revealing the interplay between data complexity and computational efficiency in predictive modeling while maintaining high precision.
- Established a comparative analysis framework, rigorously evaluating various windowing configurations and multi-output regression models using real-world serverless computing workload to ensure practical relevance and applicability.
- Analyzed the temporal stability and day-to-day performance variations of predictive models, providing insight into the reliability and operational viability of these models in serverless computing.

Journal Pre-proof

Journal Pre-proof

Optimizing Serverless Computing: A Comparative Analysis of Multi-Output Regression Models for Predictive Function Invocations

Mustafa Daraghmeh[a], Anjali Agarwal[a], Yaser Jararweh[b]

[a]*Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada*
[b]*Computer Science, Jordan University of Science and Technology, Irbid, Jordan*

Abstract

In the rapidly evolving domain of serverless computing, the need for efficient and accurate predictive methods of function invocation becomes paramount. This study introduces a comprehensive suite of innovations to improve the predictability and efficiency of function invocation within serverless architectures. By employing multi-output regression models, we perform a multi-level analysis of function invocation patterns across user, application, and function levels, revealing insights into granular workload behaviors. We rigorously investigate the impact of windowing techniques and dimensionality reduction on model performance via Principal Component Analysis (PCA), offering a nuanced understanding of data complexities and computational implications. Our novel comparative analysis framework meticulously evaluates the performance of these methods against various windowing configurations, utilizing the Azure Functions dataset for real-world applicability. In addition, we assess the temporal stability of the models and the variation of day-to-day performance, providing a holistic view of their operational viability. Our contributions address critical gaps in the predictive modeling of serverless computing and set a new benchmark for operational efficiency and data-driven decision-making in cloud environments. This study is poised to guide future advancements in serverless computing, driving theoretically sound and practically viable innovations.

Keywords: Serverless Computing, Predictive Function Invocation, Multi-Output Regression Models, Windowing Techniques, Comparative Analysis Framework

1. Introduction

Serverless computing, also known as Function-as-a-Service (FaaS), has become a cornerstone in the evolution of cloud computing [1]. FaaS is a cloud computing model in which cloud providers dynamically manage resource allocation, allowing users to run event-driven applications without the complexity of managing the underlying infrastructure [2]. This model is recognized for its efficient utilization and cost-effectiveness, offering a flexible and scalable solution for many applications [3].

The adoption of native cloud technologies, such as microservices and containers, has dramatically increased the popularity of serverless computing as an architectural choice and programming model. Detailed discussions on this topic can be found in [4] and [5], which also explore the context of serverless architecture orchestration. However, as a relatively recent innovation in the cloud computing landscape, serverless computing is still navigating its early stages of exploration and development. This growth phase aims to address critical challenges, particularly by improving the predictability and efficiency of function invocations, which are receiving more research attention [6]. Therefore, the burgeoning field of serverless computing represents fertile ground for research, especially in optimizing and refining these core aspects [7].

Serverless computing, although increasingly popular, faces various challenges, specifically in effectively handling the dynamic and unpredictable nature of function invocation [1]. The inherent unpredictability of the system adds complexity to resource management and performance optimization, which are particularly crucial in the context of High-Performance Computing (HPC) [8]. HPC entails performing computations at a higher level of performance compared to general-purpose computing. This requires the implementation of a more resilient and efficient management system in serverless environments [9]. The practice of ”warming up” functions, which involves prepping functions in advance for anticipated execution, is an approach aimed at tackling these intricacies [10].

Journal Pre-proof

This study systematically addresses the gaps in predictive function invocation within serverless computing models. By harnessing the power of data-driven modeling and simulation, our aim is to improve predictability and operational efficiency at multiple levels of the serverless architecture. We offer the following key contributions.

1. **Multi-Level Predictive Modeling:** We introduce a nuanced multi-level predictive modeling framework that utilizes multi-output regression models. This framework is designed to simulate function invocation patterns within serverless environments at the granularity of the user, application, and function levels, fostering a more precise prediction mechanism.
2. **Advanced Windowing Technique Analysis:** Through simulation, we explore the impact of varying windowing configurations on the performance of predictive models. Our comprehensive analysis of window sizes, step sizes, and target window sizes offers significant insights into their effects on the accuracy and computational load of multi-output regression models.
3. **Dimensionality Reduction and Data Complexity:** We provide a pioneering investigation into dimensionality reduction through Principal Component Analysis (PCA). This research simulates the interaction between different windowing configurations and PCA, scrutinizing its influence on the intricacies of data size and model efficacy. We propose a strategic methodology for large-scale data management that optimizes the trade-off between maintaining data integrity and enhancing computational efficiency.
4. **Comparative Analysis Framework Development:** We establish a robust comparative analysis framework that evaluates an array of inherently multi-output regression models under different windowing parameters. This framework, based on empirical data from the version 2019 Azure Functions dataset, synthesizes multiple performance metrics to identify optimal model configurations, ensuring theoretical and practical relevance.
5. **Temporal Stability and Performance Evaluation:** Our study extends beyond static model analysis by assessing the temporal stability and performance of predictive models day-to-day. This longitudinal analysis contributes to a deeper understanding of the dynamic behaviors of serverless computing models over time.

Each contribution encapsulates a strategic advancement in modeling and simulation techniques, improving predictability and efficiency in serverless computing operations. Our integrated approach advances academic research and provides actionable insights for industry practitioners.

In addition to addressing the immediate challenges of predictive function invocation in serverless computing at various pattern architecture levels, this study's contributions have broader implications for policy, practice, theory, and future research. By exploring innovative predictive methods, this research offers valuable information to policy makers in shaping effective cloud computing regulations. It also provides practical applications for cloud service providers and software developers in optimizing serverless environments. Theoretically, this work extends the current understanding of serverless computing dynamics, laying the foundation for future academic exploration. Consequently, the results of this study are poised to influence the future trajectory of serverless computing, driving advances in both the commercial and academic spheres.

The remainder of this paper is organized as follows. Section 2 presents an extensive background and related research, situating our research within the broader context of predictive modeling and serverless computing. Section 3 details the systematic methods and methodological approach used for the comparative analysis. Section 4 critically examines the results of the implemented models, discussing the results of comparative analysis and predictive methods of temporal stability at different levels. Section 5 discusses the practical implications and future directions of our research. Finally, Section 6 concludes this study with a summary of critical insights and contributions.

2. Background and Related Work

This section reviews the existing body of research on function invocation within serverless computing environments, focusing on studies that have utilized cloud workload traces for predictive analysis. The objective is to shed light on the current understanding of function invocation dynamics, evaluate the effectiveness of existing predictive models, and identify research gaps, particularly in the context of multi-level predictive analysis. This review seeks to investigate the complexities of serverless computing, with a particular focus on the necessity and potential of multi-output prediction models. This exploration is essential in order to comprehend and improve the effectiveness and accuracy of function invocation at multi-level pattern architecture, a key element of serverless computing that needs to be sufficiently examined in this rapidly advancing area.

Journal Pre-proof

2.1. Overview of Serverless Computing

Serverless computing, an innovation in cloud computing, represents a significant paradigm shift from traditional server-centric architectures to a model in which server management is abstracted. This evolution has been well documented, illustrating the transition from an emerging concept to a standard cloud service model [1, 11].

The defining characteristics of serverless computing include on-demand resource allocation and billing models, which distinguish it from conventional cloud services. In serverless architectures, resources are dynamically allocated in response to specific function calls or events, ensuring that users are billed only for the resources actually used. This approach not only offers cost benefits, but also aligns well with the elastic demand of cloud-based applications, improving power and energy efficiency in cloud data centers [1, 12].

The invocation of functions in serverless computing is central to its operational model. Functions, which are self-contained units of application logic, are executed in response to various events or triggers. This model promotes a flexible and responsive environment, making serverless computing highly suitable for event-driven applications [13, 14]. The efficiency and predictability of these invocations are crucial, as they directly impact the performance and cost-effectiveness of the serverless infrastructure [5, 15].

Furthermore, the rise of Artificial Intelligence (AI) and Machine Learning (ML) has introduced new dimensions to serverless computing. Advanced techniques such as AI-based resource allocation and adaptive auto-scaling are being integrated to optimize function invocation and resource management. This reflects the ongoing evolution and increasing sophistication of serverless architectures [16].

Serverless computing has become a transformative element in cloud computing due to its on-demand resource allocation, cost-effective billing models, and function-centric execution. The continuous development of this framework, marked by advancements in AI and ML and its growing compatibility with current computing requirements, highlights its growing importance in the cloud computing domain. The invocation of functions plays a crucial role in this model, serving as the foundation for the operational efficiency and economic viability of serverless architectures.

2.2. Function Invocation in Serverless Computing

Invocation of functions in serverless computing serves as the cornerstone of how applications respond to various events and triggers, marking a departure from traditional server-based approaches. This section explores the mechanics of function invocation, reviews research that focuses on optimization, and discusses the implications of invocation patterns on resource management and system performance.

Invocation of functions in a serverless environment is event-driven, where functions are executed in response to specific triggers such as HTTP requests, file uploads, or other cloud events. These invocations adhere to a model in which the cloud provider dynamically manages the allocation and scaling of the underlying infrastructure. The execution model, which encompasses cold starts and warm starts, significantly influences the performance and responsiveness of serverless applications [14, 17].

Optimizing the invocation process mainly revolves around reducing latency and addressing cold start issues. Cold starts occur when a function is invoked after being idle, requiring the cloud provider to allocate resources before execution. This delay can degrade the user experience. Research has explored various strategies to mitigate these delays, such as analyzing tail latency in serverless clouds [3], understanding the variability in function invocation times [18], and developing methods such as IceBreaker to improve the warming of serverless functions [10].

The patterns of function invocation have direct implications on resource management and overall system performance. Efficient invocation strategies can lead to better resource utilization, cost savings, and improved user experience. The challenge lies in accurately predicting invocation patterns and adapting resource allocation accordingly. The reviewed studies provide insight into how serverless computing can be optimized to handle varying workloads, ensuring high performance and efficient use of resources [14, 17].

2.3. Use of Cloud Workload Traces for Predictive Analysis

The use of cloud workload trace in serverless computing has become a key tool for predictive analysis, enabling a more nuanced understanding of serverless environments. This section reviews key studies that have taken advantage of various traces, discussing their methodologies, findings, and the broader implications of using real-world data in predicting serverless computing workloads.

Journal Pre-proof

Cloud workload traces, essentially records of historical server use and user request patterns, provide invaluable data for simulating and analyzing serverless computing environments. These traces offer insight into typical usage patterns, resource requirements, and performance bottlenecks, which are crucial for optimizing serverless infrastructures [19, 18].

Several studies have used cloud workload tracers for predictive analysis in serverless computing. For example, the faas-sim framework [19] uses workload traces from real-world testbeds to simulate execution time and resource usage on different computing hardware. Similarly, the research presented in [20] employs cloud workload traces to predict serverless computing workloads using advanced ML techniques like the Wasserstein Adversarial Transformer (WAT). Another study addressing the mitigation of the cold start problem in serverless computing [6] analyzes sudden spikes in workload traces and their impact on service delivery.

Using cloud workload traces for predictive analysis offers several benefits. It enables a more accurate assessment of resource needs and user demand patterns, facilitating better capacity planning and resource allocation. This approach also helps identify and address performance problems, such as latency and cold starts, by providing a real-world context for simulation and testing [21]. However, there are limitations to this approach. Analyzing the real-world workload traces at a singular pattern architecture level may only sometimes accurately represent the load patterns due to the dynamic nature of user behavior and technological advancements. In addition, privacy concerns and data security issues arise when using actual user data [22].

Ultimately, the utilization of cloud workload traces in serverless computing presents a pragmatic method to conduct predictive analysis, thus making a substantial contribution to the optimization of serverless platforms. Utilizing real-world data in serverless computing services is highly valuable for improving efficiency and responsiveness, despite the specific challenges it can pose.

2.4. Integrating Advanced Data Analytics Techniques in Prediction Models

Utilizing advanced data analytics methods like window sliding and PCA in cloud infrastructure management is essential for improving the accuracy of prediction analysis. The window sliding technique, as mentioned in references [23], [24], and [25], facilitates the division of time series data into smaller segments, which helps in improving efficiency in performance forecasting and predictive modeling. This versatile technique finds applications in diverse domains, showcasing its significance in data analysis and processing. On the other hand, PCA, as highlighted in reference [26], assists in reducing data dimensionality, improving prediction models, and enhancing visualization in cloud infrastructure management. It also simplifies complex datasets and helps identify patterns, trends, and relationships within the data, making it easier to understand and extract meaningful insights [27] [28]. Additionally, PCA can improve the performance of ML algorithms by reducing noise and multicollinearity in the data, leading to more accurate predictions and efficient models [29].

The integration of these techniques, as supported by references [30], [31], and [32], cloud-based and serverless infrastructure management can be enhanced to improve performance monitoring, capacity planning, anomaly detection, and optimized resource allocation. These techniques enable better decision-making, cost savings, and efficient resource utilization in dynamic cloud environments. As cloud technologies continue to evolve, applying such advanced analytics methods will play a vital role in ensuring the effective management of cloud resources.

2.5. Multi-Output Prediction Models

In serverless computing, where dynamic resource allocation and efficient service delivery are paramount, multi-output prediction models emerge as a critical tool for managing complex workload patterns. This section explores the concept of multi-output prediction models, their relevance in serverless computing, and how they can be leveraged to enhance function invocation processes.

Multi-output prediction models are advanced analytical tools capable of simultaneously predicting multiple dependent variables or outputs. In serverless computing, these models are particularly relevant, as they can analyze and predict various aspects of function invocation, such as invocation frequency, execution duration, and resource utilization, all of which are crucial for optimizing serverless platforms [33]. By employing these models, serverless computing systems can achieve more accurate resource provisioning, minimize latency, and improve overall service performance.

The literature reveals a growing interest in employing multi-output prediction models in cloud computing and related fields. For example, the study by Liu and Xu utilizes a Multi-output Support Vector Regression (MSVR) model

Journal Pre-proof

combined with an Immune Clone Selection Algorithm (ICSA) to improve big data in cloud computing platforms [34]. This approach underscores the potential of multi-output prediction models in handling complex, high-dimensional data common in cloud environments. Another noteworthy contribution is the TPPFaaS framework, which models serverless function invocations using Temporal Point Processes (TPPs), providing insights into workload prediction in serverless computing [33].

Multi-output prediction models in serverless computing can simultaneously predict several key aspects of function invocation. These models can forecast the frequency of invocation, helping to anticipate demand spikes and scaling resources accordingly. Additionally, they can estimate the execution duration, which is vital for managing time-sensitive tasks and reducing cold starts. Prediction of resource utilization is another critical aspect, enabling a more efficient use of computational resources and cost optimization [35].

In summary, multi-output prediction models represent a significant advancement in serverless computing, offering a comprehensive solution for predicting and managing various aspects of function invocation. By incorporating these models, serverless computing platforms can enhance their predictive capabilities, leading to more efficient, responsive, and cost-effective cloud services. The reviewed literature demonstrates the versatility and effectiveness of these models, highlighting their growing importance in the evolving landscape of cloud computing.

2.6. Gaps in Current Research

Despite the growing body of research on serverless computing, there remain significant gaps, particularly in the context of multi-level predictive analysis. This section delves into these gaps, highlighting the challenges faced by existing predictive models and underscoring the need for more advanced and nuanced approaches.

Current research in serverless computing, while extensive, often lacks a multi-dimensional approach to predictive analysis. Studies tend to focus on singular aspects of serverless environments, such as cost or performance, without fully integrating these dimensions into a cohesive predictive model. This limitation is evident in the literature, where a holistic view of serverless computing, encompassing all aspects of resource management, remains underexplored [36]. The need for comprehensive models that can simultaneously address multiple aspects of serverless computing, such as resource allocation, execution timing, and cost, is a significant gap in current research.

Existing predictive models often struggle to accurately forecast serverless computing workloads due to the highly dynamic and event-driven nature of serverless applications. These models frequently fail to account for the variability and unpredictability inherent in serverless computing environments. For example, performance modeling of serverless platforms has shown that, while simulation platforms help optimize applications, they often do not fully encapsulate the real-world complexities of serverless operations [37]. The challenge lies in creating models that are flexible and adaptable enough to handle the rapid changes in workload and resource requirements typical of serverless environments.

There is a clear need for more advanced and nuanced predictive models that can cater to the dynamic nature of serverless computing. These models should be capable of handling multi-level analysis, predicting various outcomes simultaneously, and adapting to changing conditions in real time. The literature indicates that addressing these needs will allow more effective implementation and usage of serverless computing [38]. Furthermore, according to predictions, serverless models will predominate in the future of cloud computing, underscoring the urgency of creating sophisticated predictive tools that can guarantee the effective and economical operation of these platforms [38].

In conclusion, the gaps in current research, particularly in multi-level predictive analysis, pose significant challenges to the full realization of serverless computing's potential. Addressing these gaps with more advanced predictive models at different levels is essential for the continued growth and maturation of serverless computing, enabling it to meet the evolving demands of modern cloud-based applications and services.

2.7. Summary of Reviewed Literature

The reviewed literature provides a comprehensive view of the various aspects of serverless computing, ranging from the fundamentals of function invocation to the complexities of predicting serverless workloads. This section summarizes the key points of the literature and elucidates how it informs and guides the research objectives of this study.

The following key points synthesize the core findings of the reviewed literature on serverless computing. This summary encapsulates the critical aspects that have emerged from recent research in the field, ranging from the

Journal Pre-proof

foundational concepts of serverless computing to the specific challenges and technological advancements within this domain. Each point represents a significant area of research that has contributed to the current understanding of serverless computing, its capabilities, limitations, and the direction for future innovations.

1. **Evolution and Characteristics of Serverless Computing**: The literature traces the evolution of serverless computing, emphasizing its on-demand resource allocation, billing models, and function-centric architecture. For example, the shift from traditional cloud models to serverless has fundamentally changed the way resources are managed and billed, as discussed in [1, 11].
2. **Function Invocation Dynamics**: Studies have explored the mechanics of function invocation, including triggers and execution models. A key challenge here is managing latency and cold starts, as seen in serverless platforms such as Amazon Web Services (AWS) Lambda [14, 17].
3. **Predictive Analysis Using Workload Traces**: Emphasizing the importance of cloud workload traces in predictive analysis, this point highlights their role in resource management and performance optimization. A notable example includes the use of traces to predict demand spikes and adjust resources accordingly [19, 20].
4. **Multi-Output Prediction Models**: The need for models that can predict various aspects of serverless function invocations, such as frequency and duration, is underscored. These models are crucial for optimizing resource allocation and reducing operational costs [33, 34].
5. **Gaps in Current Research and Challenges**: This point identifies the gaps in current research, particularly in multi-level predictive analysis, and discusses the limitations of existing models. It highlights the need for models that can accurately forecast dynamic serverless workloads at different levels, such as the application level for resource allocation, the function level for execution timing, and the user level for cost estimation [36, 38].

The following research objectives of this study are profoundly influenced by the insights that were gained from this extensive review of the relevant literature:

- **Understanding Serverless Computing**: By establishing a foundational understanding of serverless computing's evolution and characteristics, this study is guided towards developing predictive models at different levels that are aligned with the unique nature of serverless architectures. This understanding will inform the choice of methodologies and analysis techniques used.
- **Enhancing Prediction Accuracy**: Insights into function invocation dynamics and the effectiveness of workload traces in predictive analysis direct this research toward creating models that improve prediction accuracy, particularly in forecasting the function invocation frequency at different levels.
- **Addressing Research Gaps**: The identified gaps and challenges underscore the importance of innovating in multi-output predictive analysis. This study aims to contribute to filling these gaps by developing models that can handle the dynamic and complex nature of serverless workloads.

In conclusion, the reviewed literature not only provides a detailed understanding of the current state of serverless computing research but also clearly delineates a pathway for this study's objectives. By bridging the identified gaps and leveraging the insights from existing research, this study aims to advance the field of serverless computing through the development of improved predictive models at the user, application, and function levels.

3. Methodological Framework

3.1. Lifecycle and Invocation Patterns of Serverless Functions

Serverless computing has revolutionized the way functions are executed in the cloud, emphasizing the importance of efficient invocation patterns and lifecycle management. The performance and behavior of serverless functions are influenced by various factors, such as invocation latency, state management, workflow execution, resource allocation, and cost efficiency. An in-depth understanding of the serverless function lifecycle and its invocation patterns is paramount to optimizing these functions for better performance and cost-effectiveness in serverless architectures.

Figure 1 illustrates the standard sequence of events that occur when a serverless function is called for a specific duration. It displays the frequency of cold starts, warm starts, and subsequent keep-alive windows. The function's readiness and response latency are crucial factors that are greatly influenced by these stages, ultimately affecting the overall performance of the system. Figure 2 shows a comparative flow chart that outlines the cold start and warm start procedures. It visually represents the typical sequence of actions and transitions between execution states, providing insight into the operational intricacies of serverless functions and emphasizing the significance of optimizing start-up times by reducing cold start occurrences through proactive predictive modeling.

Research such as that presented in [3] and [39] provides valuable information on storage access patterns, invocation traffic, and the management of cold and warm starts. Such studies are critical in identifying trends and formulating strategies to reduce latency and improve resource efficiency. Furthermore, works such as [40] and [41] emphasize the importance of effective state management and workflow execution, which are essential in optimizing the performance and reliability of serverless systems.

The dynamic nature of serverless computing requires continuous monitoring and adaptation. As indicated by [42], real-time resource monitoring and predictive models are essential to effectively manage and predict resource allocations [17]. These approaches are integral to understanding the temporal dynamics of serverless functions and improving the prediction of their behavior over time.

In summary, there is a pressing requirement for sophisticated predictive models that can adjust to the distinct attributes of serverless function invocations across different levels. Through an improved understanding of these patterns and using advanced modeling techniques to predict the upcoming invocation patterns, we can vastly improve resource allocation strategies, reduce operational costs, and optimize the overall performance of serverless architectures.

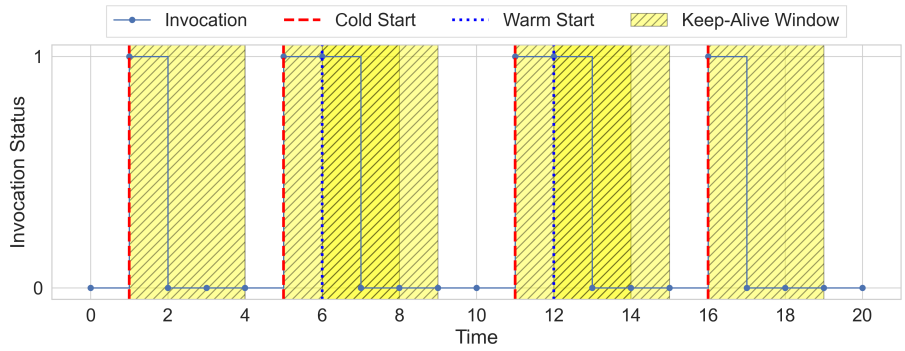
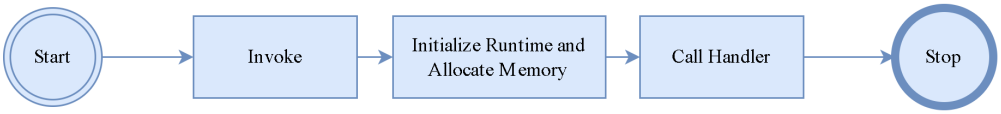


Figure 1: A Typical Serverless Function Invocation Pattern Over Time.

a) Cold Start Process: Function Invocation with Initialization



b) Warm Start Process: Function Invocation Using Preloaded Runtime

Figure 2: Comparative Flowchart of Serverless Function Invocation Processes.

Journal Pre-proof

3.2. Analyzing Function Invocation at Different Levels

This section explains how to analyze function invocation in serverless computing at different levels. The methodology covers three primary levels: the user, the application, and the function. Each level of analysis is critical for understanding the various aspects of function invocation and their impact on the serverless environment.

3.2.1. Data Preparation for Analysis Perspectives

The trace data serves as an essential component of software systems that enable analysts to comprehend the system's operations and pinpoint areas for improvement. In cloud data centers, the trace data refers to the set of records of events and activities that occur within the system, and it is systematically collected through a monitoring agent. This agent is connected to a workflow engine, which manages the sequence of tasks in a process, and a FaaS system, which allows the execution of functions in response to specific events.

In this study, the monitoring agent systematically records the rate at which invocations of functions occur over time at regular intervals. It keeps track of each function's invocation frequency within a designated time period. The raw data set of the time series $\mathcal{D}$ is represented by tuples consisting of $(h_o, h_a, h_f, T, c_1, c_2, \ldots, c_N)$ for each record. In this context, h_o represents the identifier for the user, h_a represents the identifier for the application, and h_f represents the identifier for the function. The value of T represents the type of trigger, which indicates the event that caused the function to be called. The variable c_i represents the number of times the function was called during the i^{th} time interval from the N observations within a designated period.

The raw data set $\mathcal{D}$ is manipulated to facilitate analysis from various perspectives: User, application, and function. The preparation involves aggregating the invocation counts based on different grouping criteria:

1. **User Perspective:** At the user level, we examine how users interact with the serverless platform, focusing on the frequency and patterns of function invocations. Thus, $\mathcal{D}$ is aggregated by user and trigger type (excluding h_a and h_f) to focus on usage patterns at the user level. The resulting data set D_{h_o} is defined as:

$$D_{h_o} = \sum_{h_o,T} c_i \quad \forall i \in \text{invocation counts intervals, grouped by } h_o \text{ and } T \tag{1}$$

   The summation is applied to all N function invocation counts for each unique combination of user and trigger type, with h_o subsequently excluded to emphasize trigger-based aggregation.

2. **Application Perspective:** Application-level analysis focuses on how serverless applications, as a whole, invoke functions. Aggregation is performed by application and trigger type (excluded h_o and h_f) to understand application-specific invocation patterns. The resulting data set D_{h_a} is given by:

$$D_{h_a} = \sum_{h_a,T} c_i \quad \forall i \in \text{invocation counts intervals, grouped by } h_a \text{ and } T \tag{2}$$

   Similarly to D_{h_o}, the summation is applied to all invocation counts of functions N, focusing on combinations of application and trigger type, with h_a subsequently excluded.

3. **Function Perspective:** The function level focuses on the invocation characteristics of individual functions. Function-level performance analysis involves examining the data without considering the specific user or application context. The resulting data set D_{h_f} focuses solely on trigger type and invocation counts:

$$D_{h_f} = \{T, c_1, c_2, \ldots, c_N\} \quad \text{excluding } h_o, h_a, \text{ and } h_f \tag{3}$$

This structured data preparation process transforms the raw time series data set $\mathcal{D}$ into subsets of time series data sets that facilitate the targeted analysis of user behavior, application performance, and function utilization, providing a comprehensive view of the system's operational dynamics.

However, the produced subsets time series data sets (D_{h_o}, D_{h_a}, and D_{h_f}) go through window sliding operation (detailed below) that transform the time series data into patterns that suit for the multi-output regression models. This windowing operation enables a more accurate and insightful analysis of the system's operational dynamics by transforming the time-series data into patterns suitable for our intended analysis.

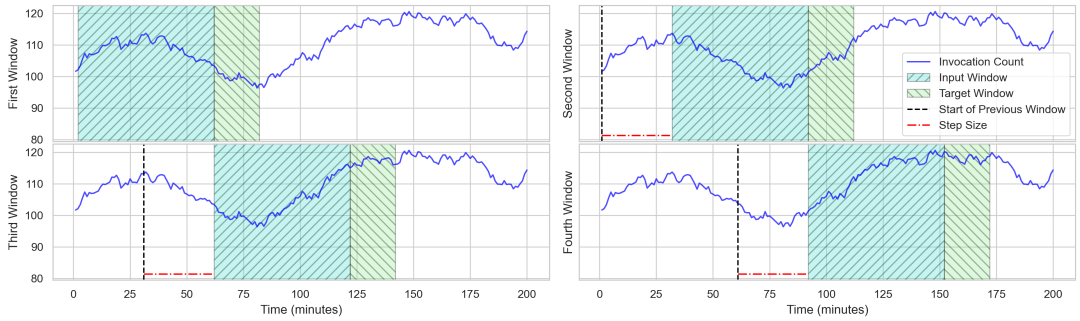


Figure 3: Sequential illustration of the windowing process applied to synthetic time series data. The figure displays four consecutive windows, with turquoise patterned areas representing input windows (z_1, z_2, z_3, z_4) and green patterned areas indicating corresponding target windows (y_1, y_2, y_3, y_4). The red dashed lines mark the initiation of new windows, underscoring the step size between them.

3.2.2. Window Sliding Operation for Invocation Time Series Data

The windowing process is a fundamental technique used in the analysis of time series data, particularly useful in ML and signal processing. Consider a discrete time series $\{x_t\}_{t=1}^{N}$ where x_t represents the value of the series at time t and N is the total number of observations. The objective of windowing is to transform this series into a set of overlapping subsequences that can be used to predict subsequent values or detect patterns.

A window, often called a frame, is defined by two parameters: the window size W_z and the step size S. The window size W_z determines the number of data points included in each subsequence, while the step size S dictates the displacement between the start of consecutive windows. The target window size W_y specifies the subsequence length we aim to predict.

Given the window size W_z, the step size S, and the target window size W_y, the i-th input window is represented as $\{x_i, x_{i+1}, ..., x_{i+W_z-1}\}$, and the corresponding target window is given by $\{x_{i+W_z}, x_{i+W_z+1}, ..., x_{i+W_z+W_y-1}\}$. The process iterates over the entire series, starting at $i = 1$ and increasing with the size of the step S to $i + W_z + W_y - 1 \leq N$.

Figure 3 illustrates the windowing process applied to synthetic data. It facilitates an intuitive grasp of how the windowing process prepares time series data for further analytical tasks. Four consecutive windows are depicted to demonstrate the time series segmentation into input and target windows. Vertical dashed lines delineate the boundaries of each window, and the patterned areas signify the input and target data points within the time series. The progression of the windowing process is visually indicated by the red dashed line, which represents the step size.

The window sliding operation, described in Algorithm 1, is crucial in converting unprocessed time series data into organized patterns appropriate for multi-output regression models. This technique is precious in serverless computing, where understanding the temporal dynamics of function invocation at different levels is vital for enhancing performance and managing resources.

The algorithm begins by initializing the lists to hold the features and targets for each window. It then iterates over the time series data, segmenting it into overlapping windows of size W_z with a step size of S and a corresponding target window of size W_y. For each iteration, it extracts the relevant feature and target windows, adding the corresponding trigger type T to enhance the data context. The final result is a pair of datasets, $X_{Pattern}$ and $y_{Pattern}$, which consist of the concatenated features and targets, respectively, with duplicates removed to ensure the uniqueness of the patterns and avoid potential biases in the model during the training process.

The application of the window sliding operation to time series data is fundamental for comprehensively analyzing serverless systems for several reasons:

- **Extraction of Temporal Features:** This technique is highly effective in extracting time-related characteristics from data, accurately capturing the patterns and timing of function invocations. Extracting this information is crucial for understanding the frequency, order, and timing of function calls, which are essential to predict the behavior of invocation at the related level and optimize performance accordingly.

Journal Pre-proof

Algorithm 1 Window Sliding Operation

Require: ts (invocation time series data), W_z (window size), S (step size), W_y (target window size)
Initialize $features\_list$, $targets\_list$
if $step\_size < 1$ **then**
    $step\_size \leftarrow 1$
end if
$R \leftarrow$ Compute range of iteration: $[ts_{start}$ to $ts_{end} - (W_z + W_y)]$ with steps of S
for $start\_slice$ in R **do**
    $end\_slice \leftarrow start\_slice + W_z$
    $target\_slice \leftarrow end\_slice + W_y$
    Extract feature and target windows from ts using $start\_slice$, end_slice, and $target\_slice$
    Add the corresponding Trigger type T into feature windows
    Append feature and target windows to $features\_list$ and $targets\_list$
end for
Concatenate $features\_list$ and $targets\_list$ into $X\_Pattern$ and $y\_Pattern$
Remove the duplicate patterns, considering combined $X\_Pattern$ and $y\_Pattern$
Ensure: Return $X\_Pattern$ and $y\_Pattern$

- **Pattern Identification and Prediction:** This method is instrumental in identifying recurrent and unique patterns in function invocation, providing a basis for detecting regularities and predicting future system states. These patterns are invaluable for forecasting workloads, identifying potential bottlenecks, and preventing system failures or anomalies. Additionally, by actively removing redundant matching patterns, this approach ensures the uniqueness and significance of the identified patterns. This refinement step improves the predictive accuracy and efficiency of the system by focusing on distinct informative patterns that contribute to a more nuanced understanding of system behavior and potential future states. It helps in maintaining a streamlined pattern database, reducing noise, and improving the speed and quality of the analytics.

- **Structural Transformation for Analytical Modeling:** By restructuring raw time series data into a sequence of overlapping windows, the operation converts variable-length sequences into a suitable consistent format for analysis. This structured approach is not only beneficial for traditional ML algorithms but is also essential for multi-output regression models, which rely on a fixed-size input to forecast multiple future points. The resultant structured data is thus more amenable to a wide variety of sophisticated analytical models, enhancing the accuracy and insight of the analysis.

- **Generic Modeling:** The sliding window technique facilitates the creation of a single generic model applicable to multiple time series, diverging from the traditional approach in which each series requires its own model. This method significantly improves scalability and flexibility. This process promotes more efficient resource utilization and enables faster adaptation to new data or changing conditions in serverless environments. By reducing the need for individual model tuning and maintenance, it streamlines the analytical process, making it more robust and adaptable to various types of time series data.

In essence, the window sliding technique transforms the subsets of time series data sets (D_{h_o}, D_{h_a}, and D_{h_f}) into patterns aligned with the analytical objectives of multi-output regression models intended in this study. This transformation is essential to dissecting and understanding the intricate operational dynamics of serverless systems, enabling stakeholders to make informed decisions based on comprehensive temporal analysis and predictions.

Studies such as [25], [36], and [43] have emphasized the importance of windowing techniques in time series analysis, particularly in the context of cloud computing, serverless, and predictive modeling. Adapting this technique to serverless computing allows a deeper understanding of the dynamics of function invocation, which is essential to optimize performance, manage resources effectively, and reduce operational costs.

In conclusion, the window sliding operation detailed in Algorithm 1 is a foundational step in the preparation of the time series datasets for in-depth analysis. Systematically segmenting the data into meaningful subsequences

Journal Pre-proof

Algorithm 2 Prepare Data for Multi-Output Regression Models

Require: $X_{Pattern}$, $y_{Pattern}$, $apply\_pca$, $variance\_threshold$, $test\_size$, $random\_state$, $objects\_path$
1: Split $X_{Pattern}$ and $y_{Pattern}$ into X_{train}, X_{test}, y_{train}, y_{test} random subsets based on $test\_size$ and $random\_state$
2: Initialize OneHotEncoder ▷ To encode categorical features as a one-hot numeric array
3: Fit OneHotEncoder on X_train and transform X_train and X_test ▷ Update X_train and X_test by replacing 'Trigger' with encoded features
4: Save OneHotEncoder object into $objects\_path$ ▷ To be used for the Hold-Out Data
5: Initialize VarianceThreshold with (threshold = 0.0) ▷ To keep all features with non-zero variance
6: Save VarianceThreshold object into $objects\_path$ ▷ To be used for the Hold-Out Data
7: Fit VarianceThreshold on X_train and transform X_train and X_test
8: **if** $apply\_pca$ is True **then**
9: Initialize $PCA\_Transformer$ with $variance\_threshold$ and $random\_state$ ▷ To keep only the first PCA components where the cumulative sum of explained variance ratio exceeds $variance\_threshold$
10: Fit $PCA\_Transformer$ on X_train and transform X_train and X_test
11: Save $PCA\_Transformer$ object into $objects\_path$ ▷ To be used for the Hold-Out Data
12: **end if**
Ensure: Return transformed X_{train}, X_{test}, y_{train}, y_{test}

Algorithm 3 Prepare Hold-Out Data for Multi-Output Regression Models

Require: $X_{holdout}$, $y_{holdout}$, $objects\_path$, $apply\_pca$
1: Load $OneHotEncoder$ from $objects\_path$
2: Transform $X_{holdout}$ using the fitted $OneHotEncoder$ ▷ Encode 'Trigger' feature as done in training
3: Load $VarianceThreshold$ from $objects\_path$
4: Transform $X_{holdout}$ using the fitted $VarianceThreshold$ ▷ Apply Variance Threshold as done in training
5: **if** $apply\_pca$ is True **then**
6: Load $PCA\_Transformer$ from $objects\_path$
7: Transform $X_{holdout}$ using the fitted $PCA\_Transformer$ ▷ Apply PCA transformation as done in training
8: **end if**
Ensure: Return transformed $X_{holdout}$, $y_{holdout}$

facilitates a range of analytical tasks and enhances our understanding of the intricate dynamics of serverless systems at user, application, and function levels, fostering a more precise prediction mechanism.

3.2.3. Invocation Pattern Preparation for Multi-Output Regression Models

For the practical application of multi-output regression models in serverless computing, it is imperative to prepare the invocation patterns that align with the predictive modeling requirements. The window sliding operation detailed in Algorithm 1 produces two key data sets: $X_{Pattern}$, representing the features of the invocation patterns, and $y_{Pattern}$, representing the corresponding target outputs over time. These data sets are further refined to fit the structure and demands of multi-output regression models. The preparation process is described methodically in Algorithm 2, which explains how to transform the invocation patterns into a suitable format for predictive modeling. This process involves a series of steps that include encoding, feature selection, and dimensionality reduction.

Initially, the invocation pattern datasets $X_{Pattern}$ and $y_{Pattern}$, derived from the window sliding operation, are split into training and testing sets based on the specified test size and random state. This division is crucial for assessing the model's performance and ensuring that it can generalize well to new data.

After splitting, OneHotEncoder is initialized to convert the categorical trigger type T feature in $X_{Pattern}$ into a one-hot numeric array, effectively transforming categorical data into a numerical format appropriate for regression analysis. The OneHotEncoder is fitted in X_{train} and then used to transform both X_{train} and X_{test}, replacing the trigger column with encoded features. This process ensures that categorical variables are appropriately handled in the model.

Subsequently, a VarianceThreshold is initialized and fitted X_{train} to remove all features with zero variance from the data, effectively streamlining the input data by eliminating redundant or noninformative variables. This step is crucial

Journal Pre-proof

Algorithm 4 Train and Evaluate Multi-Output Regression Model for Time Series Patterns

Require: $X_{Pattern}$, $y_{Pattern}$, *model*, *metric*, *data_prepare_param*, *models_path*
1: Prepare $X_{Pattern}$, $y_{Pattern}$ based on *data_prepare_param* and get X_{train}, X_{test}, y_{train}, y_{test} ▷ Using Algorithm 2
2: Train *model* on X_{train} and y_{train}
3: Save *model* into *models_path*
4: Predict with *model* on X_{test} to obtain y_{pred}
5: Replace all negative values in y_{pred} with zeros
6: Convert y_{pred} to integer
7: Calculate *score* using metric(y_{test}, y_{pred})
Ensure: Return the calculated *score*

for enhancing the model's performance by focusing on relevant features. The fitted VarianceThreshold is then used to transform both X_{train} and X_{test}, ensuring consistency in the feature space between the training and testing data sets.

If PCA is to be applied, indicated by $apply\_pca$ being True, a $PCA\_Transformer$ is initialized with the specified variance threshold and random state. This transformer is fitted to X_{train} to identify the principal components that cumulatively explain a proportion of variance that exceeds the given variance threshold. Then both X_{train} and X_{test} are transformed using this fitted $PCA\_Transformer$, reducing the dimensionality of the data while attempting to preserve as much of the original variance as possible.

Ensuring consistency and reliability in model predictions necessitates applying the same transformation process to hold-out or new incoming data as was used during training. As described in Algorithm 3, the fitted $OneHotEncoder$, $VarianceThreshold$, and $PCA_Transformer$ from the training phase are used to systematically transform the hold-out data $X_{holdout}$. This method includes encoding categorical features, removing features with zero variance, and applying dimensionality reduction, each using only the transform method of the respective object. Such a consistent application of transformations ensures that the inputs to the model maintain uniform structure and scale across training and hold-out datasets, leading to reliable and accurate predictions. This rigorous adherence to a standardized transformation process is pivotal for maintaining the integrity and efficacy of multi-output regression models, especially in the dynamic contexts of serverless computing environments.

In conclusion, Algorithms 2 and 3 systematically process the data to ensure that multi-output regression models receive input that are encoded, selected, and dimensionally reduced in a manner that maximizes their predictive performance. This preparation is fundamental to leveraging the full potential of multi-output regression models in serverless computing, providing insights into function invocation patterns and aiding in optimizing and effectively managing serverless architectures.

3.3. Adaptive Optimization Framework for Serverless Time Series Analysis

In the domain of serverless computing, understanding and optimizing the performance of multi-output regression models is crucial for effective resource management and service quality. This section outlines a comprehensive methodology for training, evaluating, and optimizing these models on time series data, particularly focusing on serverless function invocation patterns. The methodology relies on two key algorithms: Algorithm 4 for training and evaluating models and Algorithm 5 for optimizing the window sliding parameters in time series patterns.

The algorithm 4 describes the process of training and evaluating a multi-output regression model on time series patterns. Initially, the data is prepared using Algorithm 2, which ensures that $X_{Pattern}$ and $y_{Pattern}$ are appropriately transformed for the modeling process. The model is then trained on the resultant X_{train} and y_{train} datasets.

After training, the model is saved to a specified path for future inference or comparison. Predictions are made on the testing data set X_{test}, and post-processing steps are applied to ensure that the predictions are in the correct format (e.g., non-negative and integer values). Finally, the performance of the model is evaluated using a specified metric, which compares the predicted values y_{pred} with the actual values y_{test}.

The performance of time series models can often be significantly affected by the choice of parameters for window sliding. The algorithm 5 outlines an approach to optimize these parameters for a given time series data set ts. It iteratively explores combinations of window sizes W_z, step sizes S, and target window sizes W_y within the specified $window\_param\_ranges$. For each combination of parameters, the algorithm applies window sliding to the time series data to generate $X_{Pattern}$ and $y_{Pattern}$ using Algorithm 1.

Journal Pre-proof

Algorithm 5 Optimize Window Sliding Parameters for Time Series Patterns

Require: *ts*, *window_param_ranges*, *model*, *metric*, *data_prepare_param*, *optimize*

Initialize *scores* as an empty list
for each window size W_z in *window_param_ranges*['*window_size*'] **do**
  for each step size S in *window_param_ranges*['*step_size*'] **do**
    for each target window size W_y in *window_param_ranges*['*target_window_size*'] **do**
      Apply window sliding to *ts* with W_z, S, W_y and gets $X_{Pattern}$ and $y_{Pattern}$ ▷ Using Algorithm 1
      Evaluate *model* on $X_{Pattern}$, $y_{Pattern}$, & *data_prepare_param* and get *score* ▷ Using Algorithm 4
      Append ($score$, W_z, S, W_y, $model$, $data\_prepare\_param$) to *scores*
    end for
  end for
end for
if *optimize* is 'minimize' **then**
  Sort *scores* in ascending order
else
  Sort *scores* in descending order
end if
return *scores*[: 1] ▷ Return the best performing configurations

Each generated pair of feature and target data sets is then evaluated using Algorithm 4, which trains a model on the data and calculates a score representing the model's performance. These scores are collected for all combinations of parameters, and the best performing combination is determined based on whether the goal is to minimize or maximize the given performance metric.

Subsequently, the most effective arrangement and model derived from Algorithm 5 are implemented on data that has not been previously observed. This deployment entails the execution of the model on each of the subsets generated from time series data sets (D_{h_o}, D_{h_a}, and D_{h_f}), which were previously defined to represent different levels of analysis from the user, application, and function perspectives, respectively. Deploying the model is crucial to evaluate its ability to perform well and be effective in a live serverless environment, replicating real-life scenarios where the model will generate predictions on new data.

In a dynamic serverless environment, the performance of deployed models must be continuously monitored to ensure sustained effectiveness. As function invocation patterns and workload characteristics may change over time, it is critical to closely track the model's performance metrics. If a performance drop is detected, indicating a possible shift in the underlying data or its distribution, the entire process—from data preparation to model optimization—will be re-applied using the most recent data. This reapplication ensures that the model stays updated and aligned with the latest trends and changes in the serverless architecture, maintaining its accuracy and reliability. Continuous monitoring and periodic reoptimization embody a proactive approach to maintaining model performance, thereby ensuring that predictive capabilities are always tuned to the highest standard of efficiency and effectiveness.

To summarize, the comparative analysis methodology offers a strong framework for constructing multi-output regression models within the realm of serverless computing at different levels of analysis perspectives. The methodology improves the understanding and effectiveness of the models in time series data by methodically preparing the data, training the models, and optimizing the window sliding parameters. The deployment of the optimized model on unseen data provides additional assurance that the model is theoretically sound and practically effective. This approach plays a crucial role in advancing serverless computing by facilitating a more efficient allocation of resources, enhancing service quality, and improving predictive capabilities.

Moreover, combining window sliding, PCA, and multi-output regression offers a powerful approach for analyzing and modeling complex time series data, mainly when dealing with high-dimensional and potentially correlated features. This strategy uses window sliding to analyze temporal dynamics by breaking the data into localized windows. It then employs PCA to decrease the complexity of the data while preserving the necessary information and to indirectly tackle the possibility of overfitting by focusing on the features that capture the most variance. Finally, multi-output regression allows for the joint modeling of multiple interrelated outputs, providing a comprehensive understanding of

Journal Pre-proof

the relationships between features and outputs while improving prediction accuracy and interpretability. This combined framework effectively addresses the challenges of high dimensionality, temporal dynamics, and multi-output modeling, ultimately leading to more accurate, efficient, and interpretable models.

4. Experimental Results and Analysis

4.1. Azure Functions Workload

The study presented in [44] performs an in-depth analysis of the production Azure Functions workload. It provides critical insights into the characteristics and invocation frequencies of real-world functions, illuminating the operational demands placed on cloud service providers. An important observation from the study is the notably short duration of functions within the FaaS workload, particularly compared to other cloud workloads. For example, data from Azure Virtual Machine (VM) workload [45] indicates that while 63% of all VM allocations last longer than 15 minutes, less than 8% of VMs persist for 5 minutes or less. This stark contrast highlights the unique challenges of FaaS environments, which impose strict requirements on providers for rapid resource allocation and scaling. Such quick turnaround times are essential to meet the dynamic and fleeting nature of serverless function executions, necessitating efficient and agile infrastructure management.

The study also underscores the importance of comprehensively characterizing the production FaaS workload. It encompasses an array of parameters, including fundamental function types, trigger mechanisms, frequency of invocations, and the corresponding resource requirements. Addressing the scarcity of publicly available data on real-world FaaS workloads, the study emphasizes the need for detailed information on the cumulative demand faced by cloud providers. It also explores the challenges of managing cold starts and proposes a predictive policy employing time series analysis techniques such as Autoregressive Integrated Moving Average (ARIMA) modeling. This policy aims to forecast subsequent invocations and optimize resource allocation accordingly.

In the context of this study, the Azure Functions data set version 2019 presented in [44] is used, which provides granular data on the invocation counts of functions, the execution durations, and the metrics of memory usage. A specific focus is placed on analyzing the function invocation counts at different levels to discern usage patterns and operational characteristics. The data set comprises multiple files, each representing function invocations over 24 hours. Collectively, these files provide detailed insights into function usage and operational characteristics over a long period. The analysis in this study predominantly uses the first 12 days of this dataset. We employ the first day of the dataset for training and testing the model, ensuring a robust foundation for model development. The subsequent 11 days are used as hold-out data, providing a comprehensive and extended assessment of the predictive accuracy of the model. This split is crucial in the development process to avoid overfitting problems and examine the model's temporal stability. It helps guarantee that the model generalizes well, performs well in real-world settings, and is robust to potential biases and unexpected variations in the unseen function invocation patterns. However, the schema of the data set includes the following fields for each day:

- **HashOwner**: A unique identifier for the owner of the application (represented as h_o in our model).
- **HashApp**: A unique identifier for the application (represented as h_a in our model).
- **HashFunction**: A unique identifier for the function within the application (represented as h_f in our model).
- **Trigger**: Trigger that initiates the function, classified into various types (represented as T in our model).
- **[1 .. 1440]**: Columns representing the number of invocations per minute for 24 hours (represented as $(c_1, c_2, \ldots, c_N)$ in our model).

In particular, all identifiers are hashed using HMAC-SHA256 with secret salts to maintain consistency between files, enabling the correlation of data between owners, applications, and functions. The *Trigger* field categorizes the function's initiation mechanism into seven distinct groups, including HTTP, timer, event, queue, storage, orchestration, and others. The invocation fields provide a minute-by-minute account of function executions, offering a detailed view of usage patterns and demands on the Azure Functions infrastructure. This data set is invaluable for researchers and practitioners who want to optimize performance or better understand the behaviors of cloud functions.

Journal Pre-proof

4.2. Azure Functions Workload Analysis with Multi-Output Regression Models

The comprehensive study of Azure Function workload provides a pivotal foundation for applying advanced analytical methods. Notably, multi-output regression models emerge as a powerful tool to analyze function invocation patterns across various levels of granularity in serverless computing. Our proposed methodology employs these models to dissect and understand the intricate dynamics of function invocation at the user, application, and function levels, each providing unique insights into the serverless environment.

The function invocation counts derived from the Azure Functions dataset have significant value beyond their numerical representation. These figures are a critical indicator of operational efficiency and performance within a serverless architecture. As such, they are critical metrics that provide insight into the operational rhythm of the system. Therefore, it is essential to carefully monitor and analyze these metrics to optimize the performance of the serverless architecture for efficient and effective operations. By applying multi-output regression models to these counts, we can predict future invocation patterns and resource needs more accurately. This predictive capability is vital for proactive resource allocation and efficient management of serverless architectures, as it helps anticipate and mitigate potential bottlenecks and performance degradation.

The Azure Functions data set is rich and enables a comprehensive analysis at multiple levels: user level (denoted as D_{h_o} in our model), application level (denoted as D_{h_a} in our model), and function level (denoted as D_{h_f} in our model). These levels correspond to different time series data sets obtained using Equations 1, 2, and 3, respectively. Every level of analysis is indispensable to develop a comprehensive understanding of the serverless workload. Through the utilization of multi-output regression models, we conduct a detailed analysis that captures the temporal dependencies and patterns that are inherent in function invocation data. This approach enables cloud providers and system architects to make informed decisions regarding resource allocation, system scaling, and performance optimization. Our proposed methodology provides a structured approach to dissecting the serverless workload, emphasizing the adaptive and predictive aspects of modern cloud services.

4.3. Evaluation Metrics

Evaluation metrics are crucial to quantify the efficacy of predictive models. Nevertheless, the assessment of multi-output regression models necessitates metrics that can effectively measure the precision of predictions across multiple outputs. We used a set of metrics, implemented through the scikit-learn Python library [46], to guarantee a detailed examination of model accuracy and dependability in our research.

4.3.1. Mean Absolute Error

Mean Absolute Error (MAE) is a widely used metric in statistical analysis to evaluate the precision of predictive models. It is a scale-dependent accuracy measure that is used to quantify the prediction errors of a regression model. It is defined as the average of the absolute differences between the predicted values and the actual values. For multi-output regression problems, MAE is computed for each output separately, and an average over all outputs can be used to obtain a single performance measure. The MAE for a multi-output regression is given by:

$$MAE = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{n}\sum_{i=1}^{n}|y_{ij} - \hat{y}_{ij}|, \tag{4}$$

where y_{ij} is the actual value of the j-th output for the i-th sample, $\hat{y}_{ij}$ is the corresponding predicted value, n is the number of samples, and m is the number of outputs. In scikit-learn python library, this metric is implemented as `sklearn.metrics.mean_absolute_error` and can be used to evaluate each output independently or in aggregate. MAE is particularly useful because it provides a straightforward measure of predictive accuracy with a clear physical interpretation, as it is expressed in the same units as the data [47], and smaller values are considered better.

4.3.2. Mean Squared Error

Mean Squared Error (MSE) is particularly useful in highlighting larger errors due to its quadratic nature. For multi-output regression models, MSE is computed for each output variable, and similarly to the MAE metric, an average MSE can be used as an overall measure of model performance. The MSE for multi-output regression is:

Journal Pre-proof

$$MSE = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{n}\sum_{i=1}^{n}(y_{ij} - \hat{y}_{ij})^2. \quad (5)$$

Scikit-learn provides this metric as `sklearn.metrics.mean_squared_error`. This metric is especially valuable when large errors are particularly undesirable in the multi-output regression context. It is sensitive to outliers and provides a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better [48].

4.3.3. Coefficient of Determination

The coefficient of determination, denoted R^2, is a key metric in regression analysis that measures the proportion of variance in the dependent variable that is predictable from the independent variable(s). For multi-output regression problems, it reflects the proportion of variance for each dependent variable that is explained by the independent variables in the model. The overall R^2 can be calculated as the unweighted mean of the R^2 values for each output, which is the approach taken by scikit-learn's `sklearn.metrics.r2_score` function, which is computed as:

$$R^2 = \frac{1}{m}\sum_{j=1}^{m}(1 - \frac{\sum_{i=1}^{n}(y_{ij} - \hat{y}_{ij})^2}{\sum_{i=1}^{n}(y_{ij} - \bar{y}_j)^2}), \quad (6)$$

where $\bar{y}_j$ is the mean of the j-th output. It is crucial to note that this metric can be sensitive to the number of predictors in the model, which should be considered when evaluating models with a different number of independent variables. R^2 values range from 0 to 1, with higher values indicating a better fit of the model to the data [49].

A multifaceted evaluation of model performance is crucial for ensuring the generalizability and robustness of the proposed models. To achieve this, we use a combination of evaluation metrics to extend beyond the sole consideration of average accuracy and dive deeper into the specific characteristics of the models. The MAE provides valuable information on the typical error behavior of the model by capturing the average magnitude of errors. Although it does not heavily penalize large errors, it offers a good understanding of the general performance of the model. The R^2 focuses on the ability of the model to explain the variance in the target output, highlighting its explanatory power. This metric goes beyond simply assessing the model's ability to predict the correct value and evaluates its capacity to capture the underlying relationships within the data. Finally, the MSE incorporates the magnitude of errors by squaring them, providing insight into the model's sensitivity to significant deviations from the predicted value. By incorporating all three metrics, we comprehensively understand the strengths and weaknesses of the models, encompassing both their average performance and their sensitivity to extreme errors and data variability. This multifaceted approach enables informed decision-making when selecting the most suitable model for specific needs, ensuring generalizability and robustness in real-world applications.

4.4. Experimental Setup

In the context of our adaptive optimization framework, the selection of appropriate window parameters is critical for an effective time-series analysis of serverless function invocations. The `window_param_ranges` dictionary defines the range of values over which the window sliding parameters will be optimized. Specifically, it includes:

- **Window size** (W_z): This parameter determines the length of each window or segment of the time series data that will be considered for analysis. In our study, the window size is varied among [30, 60, 90, 120] minutes, allowing us to understand how the choice of window size impacts the model's ability to capture relevant patterns in function invocations.
- **Step size** (S): This parameter specifies the step or displacement between consecutive windows. A smaller step size means higher overlap between consecutive windows and more fine-grained analysis, while a larger step size reduces the computational load at the expense of granularity. Our step sizes are set at [1, 15, 30] minutes, ensuring a range from high overlap to moderate overlap.
- **Target window size** (W_y): This refers to the size of the window to predict future values. It is essential to determine how far ahead the model should forecast. We consider target window sizes of [30, 60] minutes to explore short-term predictions within the scope of the behavior of serverless functions at each level.

Journal Pre-proof

By iterating over these ranges, the optimization algorithm 5 systematically explores various configurations of window sliding, assessing which combination best captures the temporal dynamics of serverless function invocations while balancing the trade-off between computational efficiency and predictive accuracy.

In our adaptive optimization framework, the data preparation phase is instrumental in transforming the raw time series data into a suitable format for multi-output regression modeling. The `data_prepare_param` dictionary encapsulates the key parameters that guide this phase, ensuring that each data set is appropriately processed before being fed into the model. Specifically, it includes:

- *apply_pca*: The parameter is set to either [False, True], indicating whether PCA should be applied or not. In this study, we examine both cases in which applying PCA can mitigate the curse of dimensionality and improve the performance of the model by focusing on the components that account for the most variance in the data.
- **variance_threshold**: This parameter, set at 0.99, defines the amount of variance that needs to be captured by the selected components in PCA. A high threshold like 0.99 means that the model will attempt to retain the components that together explain 99% of the variance, ensuring that most of the original information is preserved while still benefiting from reduced feature space.
- **test_size**: This parameter dictates the proportion of the data set to be reserved for testing. Here, a value of 0.2 indicates that 20% of the data will be used as a test set while the remaining 80% will constitute the training set. This split is crucial for evaluating the model's performance, ensuring it generalizes well to unseen data, and preventing overfitting.

By adjusting these parameters, researchers and practitioners can fine-tune the data preparation process to suit the specific needs and constraints of their analytical tasks. The `data_prepare_param` dictionary therefore plays a vital role in setting up the data for subsequent modeling and analysis, directly impacting the efficiency and effectiveness of the predictive models.

In our quest to identify the most effective windowing and data preparation configurations for serverless time series, we have used the Linear Regression (LR) model from the Scikit-learn library as a foundational modeling technique. LR model is inherently capable of multi-output regression, making it particularly suited for our scenario where predicting multiple future invocation counts is necessary.

We have opted for the default configuration of the LR model provided by Scikit-learn. By leveraging the simplicity and effectiveness of this model, we aim to dissect and understand the relationship between the windowing parameters and the predictive accuracy of our models. This approach allows us to systematically explore and optimize the window size, step size, target window size, and data preparation configurations, ensuring that our final setup is well tuned to provide accurate and timely predictions for function invocations in serverless computing environments.

Building on the best configuration identified in the preceding step, this study employs, in addition to LR, a suite of regression models for in-depth analysis, each initialized with its default settings as specified by Scikit-learn (version 1.2.2). These models are inherently designed for multi-output regression tasks, encompassing K-Neighbors (KNN), Decision Tree (DT), Random Forest (RF), and Extra Trees (ET) regressors. Each model has been carefully selected for its suitability to handle complex multidimensional output without further implementation, providing a robust analytical framework. The specific parameters and their respective default values for each model are meticulously documented in Table 1, providing a comprehensive overview of the models' configurations used in this study. This methodology guarantees a consistent and adaptable analytical setting, facilitating accurate and reliable analysis.

4.5. Comparative Analysis of Window Sliding Parameters

This section delves into a comprehensive comparative analysis of window sliding parameters at different levels of function invocation within serverless computing environments with and without the application of dimensionality reduction. By dissecting the user, application, and function levels, we understand how predictive modeling can be optimized across various aspects of serverless architecture. The use of LR in this comparison allows a straightforward interpretation of the relationship between input features and predicted results, making it a suitable choice to understand the impact of window sliding parameters on model performance. In addition, it reduces the computation costs associated with more complex models.

Journal Pre-proof

Table 1: Detailed Parameters for Inherently Multi-Output Regression Models

Index	Model	Parameters
LR	Linear Regression	copy_X: True, fit_intercept: True, positive: False
KNN	K-Neighbors Regressor	algorithm: auto, leaf_size: 30, metric: minkowski, metric_params: None, n_neighbors: 5, p: 2, weights: uniform
DT	Decision Tree Regressor	ccp_alpha: 0.0, criterion: squared_error, max_depth: None, max_features: None, max_leaf_nodes: None, min_impurity_decrease: 0.0, min_samples_leaf: 1, min_samples_split: 2, min_weight_fraction_leaf: 0.0, random_state: 123, splitter: best
RF	Random Forest Regressor	bootstrap: True, ccp_alpha: 0.0, criterion: squared_error, max_depth: None, max_features: 1.0, max_leaf_nodes: None, max_samples: None, min_impurity_decrease: 0.0, min_samples_leaf: 1, min_samples_split: 2, min_weight_fraction_leaf: 0.0, n_estimators: 100, oob_score: False, random_state: 123, warm_start: False
ET	Extra Trees Regressor	bootstrap: False, ccp_alpha: 0.0, criterion: squared_error, max_depth: None, max_features: 1.0, max_leaf_nodes: None, max_samples: None, min_impurity_decrease: 0.0, min_samples_leaf: 1, min_samples_split: 2, min_weight_fraction_leaf: 0.0, n_estimators: 100, oob_score: False, random_state: 123, warm_start: False

The analysis is presented methodically through Tables 2, 3, and 4, each illustrating the implications of employing PCA on the predictive precision. Furthermore, columns P_c and P'_c denote the original and reduced counts of data points, respectively, illustrating the volume of data being processed and the resulting reduction by removing the redantant patterns according to the given window sliding parameters.

4.5.1. User-Level Analysis

The user level analysis, shown in Table 2, is critical for understanding how individual behavior impacts the invocation of serverless functions and thereby influences the overall system's performance. By examining the window sliding parameters in invocation time series data at the User level, we aim to optimize predictive models that can accurately forecast user interactions with serverless functions. The analysis is classified based on the application of PCA, a technique crucial in managing the dimensionality of the data, thus affecting the complexity and efficiency of the predictive model.

Without PCA (Table 2a): The model without PCA provides a baseline utilizing the full spectrum of data dimensions. This approach offers a detailed view of user interactions but at a higher computational cost. The highest performing configuration, as indicated by the lowest MAE value of **22.4001**, is observed with a window size (W_z) of **120**, a step size (S) of **1**, and a target window size (W_y) of **60**. Furthermore, the volume of patterns that are processed (P_C) of **19101628** and the resulting reduction (P'_c) of **7307047** with a reduction percentage of around **61.74%** due to the large number of redundant patterns. This outcome indicates that a finer temporal resolution and a more extensive historical context lead to more accurate predictions at the user level. However, as we move down the table, increasing S and decreasing W_z or W_y generally correspond to an increase in MAE, suggesting a loss of critical temporal information or context, which is detrimental to model performance.

With PCA (Table 2b): The PCA-applied model aims to reduce the computational load by simplifying the data feature set while attempting to retain the most significant variance within the data. The optimal configuration under PCA shows a MAE of **21.7611**, achieved with a window size (W_z) of **120**, a step size (S) of **30**, and a target window size (W_y) of **30**. Also, the volume of patterns that are processed (P_C) of **666512** and the resulting reduction (P'_c) of **256037** with a reduction percentage of around **61.74%**. Interestingly, this configuration, despite the reduced dimensionality and volume of the pattern, outperforms the best non-PCA model in terms of MAE, underscoring the effectiveness of PCA in enhancing model performance by focusing on the most influential data features and the

Journal Pre-proof

Table 2: Comparative Analysis of Window Sliding Parameters in Invocation Time Series Data at User Level.

(a) PCA = False

MAE	W_z	S	W_y	P_c	P'_c
22.4001	120	1	60	19101628	7307047
22.6566	120	1	30	19556068	7043090
23.4433	120	30	30	666512	256037
23.6819	120	15	30	1317876	499528
23.7114	120	15	60	1287580	516113
24.1967	90	30	30	681660	244458
24.2066	90	1	60	19556068	7043090
24.6357	90	1	30	20010508	6669682
24.8298	90	15	60	1317876	499528
24.8914	90	30	60	666512	256037
24.9393	90	15	30	1348172	476038
25.4298	120	30	60	651364	264341
26.1004	60	1	60	20010508	6669682
26.1917	60	30	60	681660	244458
26.4095	60	15	60	1348172	476038
26.5360	60	30	30	696808	227051
26.9603	60	15	30	1378468	440733
27.7661	60	1	30	20464948	6119737
29.2018	30	15	60	1378468	440733
29.3698	30	30	60	696808	227051
30.0785	30	1	60	20464948	6119737
30.9767	30	15	30	1408764	390280
32.3219	30	30	30	711956	201494
34.1617	30	1	30	20919388	5384934

(b) PCA = True

MAE	W_z	S	W_y	P_c	P'_c
21.7611	120	30	30	666512	256037
22.5365	120	15	60	1287580	516113
22.7309	120	15	30	1317876	499528
22.9182	90	30	30	681660	244458
23.2235	120	1	60	19101628	7307047
23.4465	90	30	60	666512	256037
23.5943	120	30	60	651364	264341
23.6823	120	1	30	19556068	7043090
23.7199	90	15	60	1317876	499528
24.1475	90	15	30	1348172	476038
24.9023	60	30	60	681660	244458
25.4551	90	1	60	19556068	7043090
25.5474	60	30	30	696808	227051
25.6770	60	15	60	1348172	476038
26.0935	90	1	30	20010508	6669682
26.5803	60	15	30	1378468	440733
27.7015	60	1	60	20010508	6669682
29.0088	30	30	60	696808	227051
29.3909	30	15	60	1378468	440733
29.7438	60	1	30	20464948	6119737
31.1230	30	1	60	20464948	6119737
31.3388	30	30	30	711956	201494
31.4411	30	15	30	1408764	390280
35.1279	30	1	30	20919388	5384934

characteristic uniqueness of the patterns. However, the trade-off between dimensionality reduction and loss of detail is evident across various configurations, emphasizing the need for a balanced approach in model construction and feature selection.

Implications of Findings: The detailed comparative analysis at the user level reveals several important implications. First, the choice of window sliding parameters has a profound impact on the predictive accuracy of the models, with larger window sizes generally providing more context for prediction but requiring careful consideration of the step and target sizes to maintain model performance. Second, applying PCA can significantly improve computational efficiency and, in some configurations, even improve predictive precision by eliminating redundant information. However, its application must be judicious, ensuring that the reduction in dimensionality does not overlook critical behavioral patterns essential for accurate prediction. Lastly, the sorted results based on MAE provide a clear hierarchy of the performance of the model, guiding the selection of the appropriate configurations based on the specific needs for accuracy and computational resources.

Future Directions: Moving forward, further research could explore the dynamic adaptation of window sliding parameters and PCA components based on user behavior patterns, considering segmentation as a prior process like the work presented in [50, 51]. This process could lead to more flexible and accurate predictive models capable of adapting to the evolving nature of user interactions in serverless environments. Furthermore, investigating hybrid models that integrate multi-level learning strategies and cost estimation techniques, such as those introduced in [52] and [53] respectively, could provide novel avenues for enhancing the predictability and cost-effectiveness of serverless computing systems at the user level.

4.5.2. Application-Level Analysis

Table 3 shows a comprehensive comparative analysis of window sliding parameters at the application level. The focus shifts to how applications comprising multiple functions interact within the serverless environment, both with and without the application of dimensionality reduction. Accordingly, we can dissect and understand the collective behavior of functions as they interact within specific applications. This level is crucial for understanding and optimizing resource allocation, scalability, and overall performance of serverless systems.

Without PCA (Table 3a): The application-level analysis without PCA serves as a complete representation of the complexity of all the data. The configuration with the highest performance in terms of predictive precision is marked by a MAE of **18.4511**, achieved with a window size (W_z) of **120**, a step size (S) of **1**, and a target window size (W_y) of **60**. This configuration reflects that higher data degrades computational efficiency. As the table progresses, the variations in W_z, S, and W_y show the corresponding changes in the MAE, indicating the sensitivity of the model performance to these parameters. In the best-performing configuration, the count of original data points (P_c) is **27700387**, while the reduced data points (P'_c) is **9274343**. This reduction shows that the windowing approach is also highly efficient in managing the volume of data at the application level, reducing the size by around **66.51%**. This decrease in data points leads to a more streamlined predictive process.

With PCA (Table 3b): When PCA is applied, the analysis progresses toward understanding how dimensionality reduction affects predictive modeling at the application level. The most effective configuration results in a reduced MAE of **16.9978**, indicating an improvement in predictive precision. This configuration is achieved with a window size (W_z) of **120**, a step size (S) of **30**, and a target window size (W_y) of **30**. The reduction in data volume is also notable, with PCA further reducing the complexity of the data set while maintaining predictive quality. The results indicate that PCA effectively balances the need for computational efficiency with the requirement of accurate predictions at the application level. In the most optimal setup, the number of original data points (P_c) is **944581**, whereas the reduced data points (P'_c) amount to **338099**. This decrease demonstrates that the windowing technique is also extremely effective in handling the amount of data at the application level when PCA is applied, resulting in a reduction of approximately **64.20%** in size.

Implications of Findings: The comparative analysis at the application level underscores several key points. First, the choice of window sliding parameters significantly influences predictive performance, with larger windows providing more historical context but also requiring more computational resources. Second, PCA's role in reducing dimensionality is beneficial in terms of computational efficiency and can lead to improved accuracy in certain configurations. However, the delicate balance between data reduction and the preservation of essential information is crucial. The results offer a guide for selecting the right configuration based on specific needs with respect to accuracy and computational constraints.

Future Directions: Future research might explore adaptive window sliding and dimensionality reduction techniques, perhaps exploring how these parameters can be dynamically adjusted based on changing patterns of application usage. Similarly to the user level, further investigation of hybrid and advanced modeling techniques could also provide more nuanced insights into predictive accuracy and computational efficiency at the application level. Additionally, incorporating real-time learning and update mechanisms into these models could improve their adaptability and responsiveness to evolving application behaviors and requirements.

Studying the sliding parameters of the window at the application level provides a thorough understanding of how different configurations impact predictive modeling in serverless computing. It provides a means to enhance these models, guaranteeing they are both feasible and effective in their predictive abilities.

4.5.3. Function-Level Analysis

The function level represents the most granular aspect of serverless computing, focusing on individual function invocations. This level of analysis, shown in Table 4, is essential for fine-tuning the performance of serverless systems at the most fundamental level. It is crucial to understand and optimize the invocation patterns of individual functions, which are the fundamental units of execution in serverless architectures. The comparative analysis includes scenarios with and without the application of dimensionality reduction, underscored the sensitivity of function invocations to windowing parameters, with distinct patterns emerging across different configurations.

Without PCA (Table 4a): At the function level, the non-PCA model delivers its best performance with an MAE of **12.3496**, utilizing a window size (W_z) of **120**, step size (S) of **15**, and target window size (W_y) of **30**. This setup indicates that, at the function level, a moderately sizeable historical context with a more considerable step size

Journal Pre-proof

Table 3: Comparative Analysis of Window Sliding Parameters in Invocation Time Series Data at Application Level.

(a) PCA = False

MAE	W_z	S	W_y	P_c	P'_c
18.4511	120	1	60	27700387	9274343
18.6181	120	15	30	1911129	635095
18.6752	120	30	60	944581	338099
18.7141	120	1	30	28359397	8892705
19.3863	120	15	60	1867195	659296
19.5960	120	30	30	966548	325965
19.6540	90	15	60	1911129	635095
20.0009	90	1	60	28359397	8892705
20.5056	90	1	30	29018407	8365574
20.6778	90	15	30	1955063	601542
20.7204	90	30	30	988515	309345
20.7650	90	30	60	966548	325965
21.2986	60	15	60	1955063	601542
21.6550	60	30	30	1010482	284899
21.7345	60	1	60	29018407	8365574
22.0113	60	30	60	988515	309345
22.7276	60	15	30	1998997	552107
23.0547	30	30	60	1010482	284899
23.0890	60	1	30	29677417	7608574
24.4084	30	15	60	1998997	552107
24.7596	30	1	60	29677417	7608574
26.9295	30	30	30	1032449	250048
27.1122	30	15	30	2042931	483687
28.6774	30	1	30	30336427	6624745

(b) PCA = True

MAE	W_z	S	W_y	P_c	P'_c
16.9978	120	30	60	944581	338099
17.2195	120	15	30	1911129	635095
17.8888	120	30	30	966548	325965
18.1780	120	15	60	1867195	659296
18.7491	90	15	60	1911129	635095
19.0811	120	1	60	27700387	9274343
19.3624	90	30	60	966548	325965
19.4455	90	30	30	988515	309345
19.5242	120	1	30	28359397	8892705
19.7444	90	15	30	1955063	601542
20.6265	60	30	30	1010482	284899
20.6860	60	15	60	1955063	601542
20.7000	60	30	60	988515	309345
20.7488	90	1	60	28359397	8892705
21.4177	90	1	30	29018407	8365574
22.2539	60	15	30	1998997	552107
22.5325	30	30	60	1010482	284899
22.8589	60	1	60	29018407	8365574
24.5272	60	1	30	29677417	7608574
24.5407	30	15	60	1998997	552107
25.5793	30	1	60	29677417	7608574
26.4422	30	30	30	1032449	250048
27.3422	30	15	30	2042931	483687
29.3699	30	1	30	30336427	6624745

effectively captures the necessary temporal information for accurate predictions. The original (P_c) is **4037844** and the reduced data points (P'_c) is **964653**, which reflect the volume of processed data with a notable reduction around **76.10%** indicating the efficiency of the windowing parameters chosen to condense the data while preserving essential information for prediction.

With PCA With PCA (Table 4b): Implementing PCA at the function level, the optimal configuration improves the MAE to **11.7049**. This performance is achieved with similar window sliding parameters as the optimal non-PCA configuration, demonstrating that PCA can enhance the model's predictive accuracy by concentrating on the most significant aspects of the data. The reduction in data volume is consistent with the non-PCA configuration, with PCA contributing to a more manageable and efficient predictive modeling process while ensuring the quality of predictions is improved.

Implications of Findings: The function-level comparative analysis sheds light on the importance of carefully selecting window sliding parameters to optimize predictive models in serverless environments. It demonstrates that larger window sizes can provide more historical context for predictions, but need to be balanced with the computational costs associated with processing more extensive data. The application of PCA shows promise in reducing these computational demands while maintaining or even enhancing predictive accuracy in certain configurations. However, it also highlights the need to carefully consider the amount of dimensionality reduction to ensure that critical information is not lost.

Future Directions: Further research could explore adaptive and dynamic techniques for window sliding and dimensionality reduction at the function level, tailored to the unique characteristics and usage patterns of individual serverless functions. Additionally, investigating the integration of real-time data streams and incremental learning approaches could provide more accurate and up-to-date predictions. Such advancements would contribute to more

Journal Pre-proof

Table 4: Comparative Analysis of Window Sliding Parameters in Invocation Time Series Data at Function Level.

(a) PCA = False

MAE	W_z	S	W_y	P_c	P'_c
12.3496	120	15	30	4037844	964653
12.5688	120	1	60	58525532	14108133
12.7044	120	1	30	59917892	13300193
13.1069	90	15	60	4037844	964653
13.4316	120	30	60	1995716	524678
13.5113	90	1	60	59917892	13300193
13.6413	120	15	60	3945020	1018527
13.8290	120	30	30	2042128	497655
14.2297	90	15	30	4130668	895434
14.3957	90	1	30	61310252	12259593
14.6203	60	15	60	4130668	895434
14.6787	90	30	60	2042128	497655
14.7083	90	30	30	2088540	462987
15.1948	60	30	60	2088540	462987
15.2487	60	1	60	61310252	12259593
15.7330	60	15	30	4223492	802729
16.2441	60	30	30	2134952	416190
16.6449	60	1	30	62702612	10892928
16.7448	30	15	60	4223492	802729
17.0246	30	30	60	2134952	416190
17.8540	30	1	60	62702612	10892928
19.2159	30	30	30	2181364	354307
19.5070	30	15	30	4316316	683458
21.3097	30	1	30	64094972	9220859

(b) PCA = True

MAE	W_z	S	W_y	P_c	P'_c
11.7049	120	15	30	4037844	964653
12.1630	120	30	60	1995716	524678
12.6766	120	15	60	3945020	1018527
12.7933	120	30	30	2042128	497655
12.8108	90	15	60	4037844	964653
12.9151	120	1	60	58525532	14108133
13.1676	120	1	30	59917892	13300193
13.6337	90	30	60	2042128	497655
13.6620	90	30	30	2088540	462987
13.9057	90	15	30	4130668	895434
14.0573	90	1	60	59917892	13300193
14.2341	60	30	60	2088540	462987
14.4291	60	15	60	4130668	895434
15.0695	90	1	30	61310252	12259593
15.3295	60	30	30	2134952	416190
15.6768	60	15	30	4223492	802729
15.8668	60	1	60	61310252	12259593
16.4722	30	30	60	2134952	416190
16.6106	30	15	60	4223492	802729
17.4837	60	1	30	62702612	10892928
18.1948	30	1	60	62702612	10892928
18.7374	30	30	30	2181364	354307
19.2839	30	15	30	4316316	683458
21.6151	30	1	30	64094972	9220859

responsive and efficient serverless architectures that are capable of adapting to the evolving demands and behaviors of applications and users.

This function-level analysis is a critical component in understanding and optimizing serverless computing systems. By providing a nuanced view of the impact of window sliding parameters and dimensionality reduction techniques, this analysis contributes to the development of more sophisticated and effective predictive models for serverless function invocations.

4.5.4. Comparative Summary

The detailed findings from these comparative analyses provide a wealth of information on the design and optimization of predictive models in serverless computing environments. At each level of analysis, as shown in Figures 4 to 9, the interaction between the windowing parameters and PCA highlights the multifaceted nature of predictive modeling, where the choice of parameters significantly influences the effectiveness of the model. Furthermore, the variation in MAE across different configurations sheds light on the inherent complexities of predicting serverless function invocations, underscoring the need for tailored approaches that take into account the unique characteristics of user, application, and function behaviors.

The heatmaps presented in Figures 4, 5, and 6 provide a visual representation of the MAE for a LR model across different levels of analysis (user, application, and function) with varying window sizes (W_z), target sizes (W_y), and step sizes (S). The color gradients within the heatmaps range from blue to red, indicating low to high MAE values, respectively.

At the User level, the heatmaps contrast the effects of window sliding parameters on the MAE with and without PCA application. The heatmap without PCA shows that the lowest MAE is obtained with a larger W_z and smaller

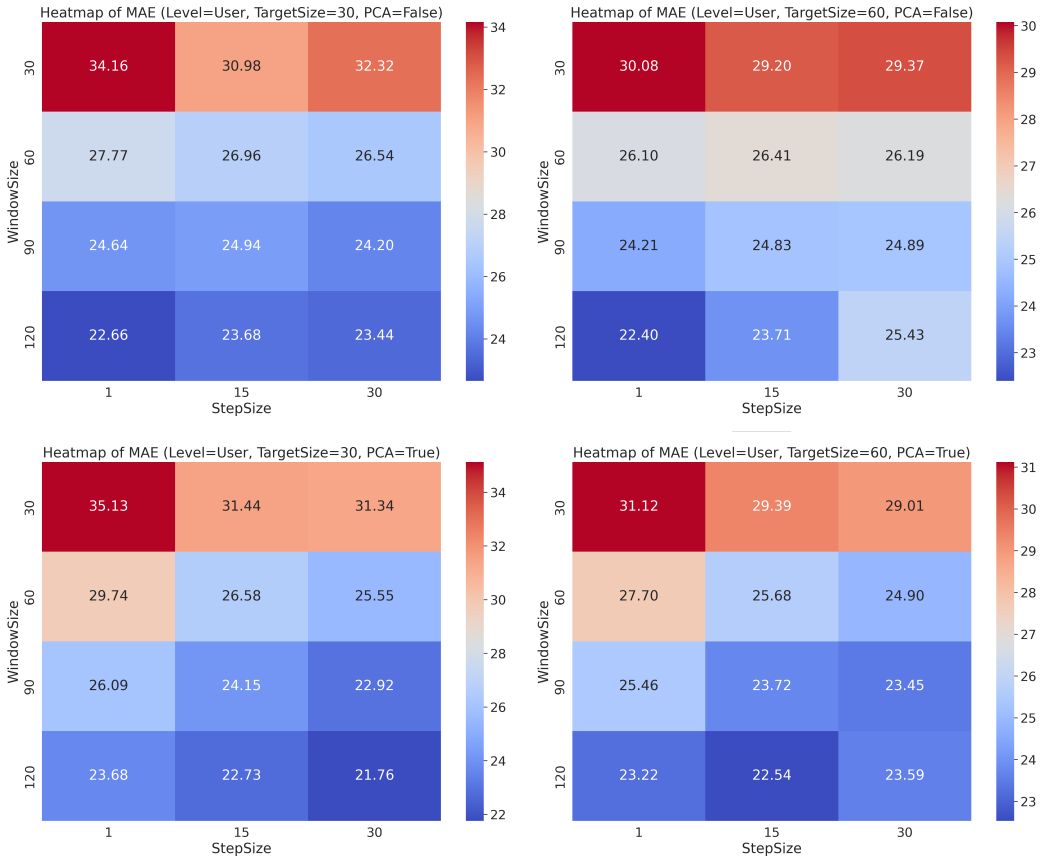


Figure 4: Comparative Heatmaps of Mean Absolute Error for Linear Regression Model: Effects of Window Size, Target Size, and Step Size at User level With and Without the application of PCA.

step size, suggesting that capturing finer temporal resolutions is crucial at this level. The application of PCA, as seen in the corresponding heatmap, generally results in lower MAE values across all configurations, demonstrating the effectiveness of dimensionality reduction in enhancing model performance by filtering out noise and less relevant features.

For the application level, the heatmap without PCA indicates that a larger historical context (larger W_z) tends to improve prediction accuracy, as lower MAE values are observed. Upon applying PCA, the heatmap shows an overall improvement in MAE, with the most significant reduction achieved with a W_z of **120** and a step size of **30**. This result highlights PCA's role in distilling critical features for improved predictive accuracy.

The function level analysis heatmap without PCA suggests that a balance between W_z and step size is key to accurate predictions, with the lowest MAE recorded for an intermediate step size of 15. The introduction of PCA leads to an even lower range of MAE values, reinforcing the premise that dimensionality reduction, which focuses on preserving significant variance, is beneficial for model accuracy at the function level.

Across all levels, the introduction of PCA consistently enhances model accuracy, as evidenced by the cooler color tones in the heatmaps. Larger window sizes are typically associated with better performance, indicating the value of extensive historical data for the predictive models. The step size needs to be optimized to avoid missing critical temporal patterns or failing to capture sufficient data variability.

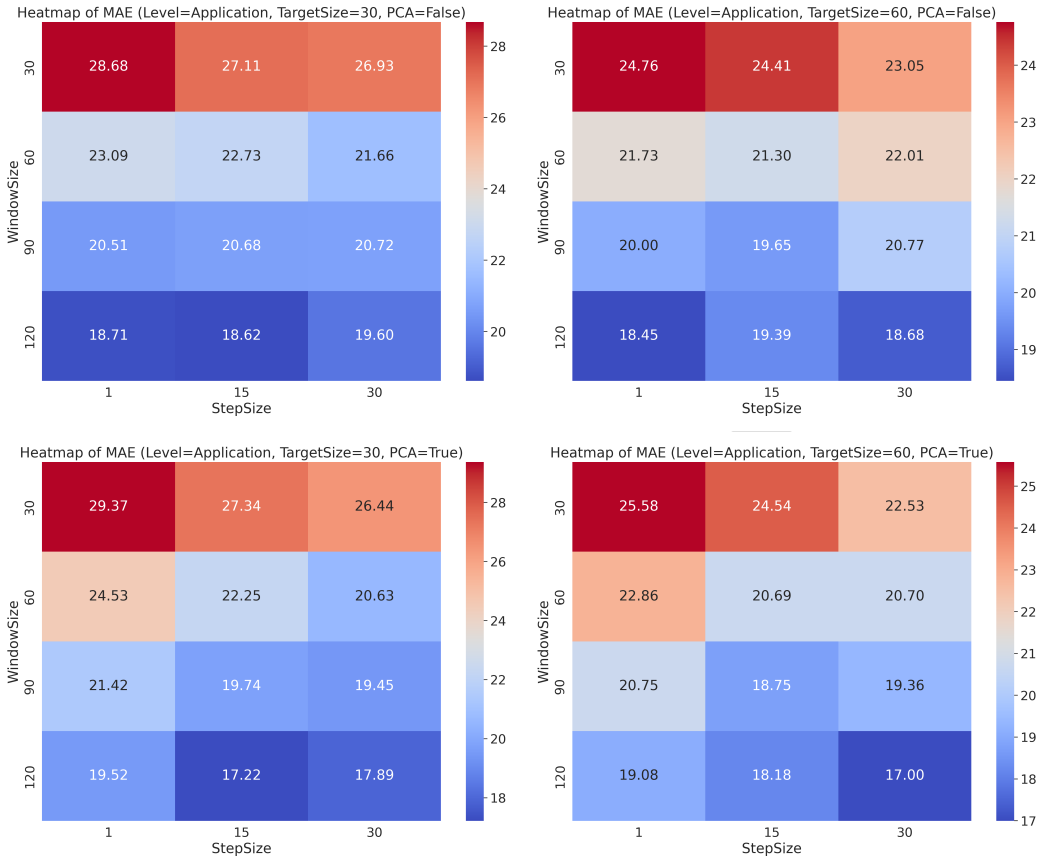


Figure 5: Comparative Heatmaps of Mean Absolute Error for Linear Regression Model: Effects of Window Size, Target Size, and Step Size at Application level With and Without the application of PCA.

The heatmaps effectively guide the selection of optimal window sliding parameters for serverless computing predictive models. They emphasize the trade-off between the capture of detailed temporal information and the computational gains achieved through dimensionality reduction techniques such as PCA.

Figures 7, 8, and 9 provide a comprehensive comparison of pattern analysis at multiple levels, highlighting the correlation between raw (P_c) and filtered (P'_c) pattern counts, MAE, and the variations in windowing parameters with and without the application of PCA.

A critical aspect of this analysis is the emphasis on reducing pattern counts. The bar graphs illustrate the comparison between raw pattern counts and filtered pattern counts after redundant patterns are removed. This reduction is crucial as it directly influences the MAE, with a significant reduction in patterns often correlating with improved predictive accuracy, as indicated by the line graphs showing the MAE trend.

At the User level, the graphs reveal that without PCA, while there is a substantial reduction in pattern count, the MAE remains relatively high. However, with PCA, not only does the pattern count reduction remain significant, but the MAE also decreases, underscoring PCA's role in enhancing model performance by focusing on the most informative features.

Similarly, the application-level graphs demonstrate a notable reduction in pattern count. It is observed that the application of PCA contributes to a further reduction in MAE, indicating an efficient balance between data simplification

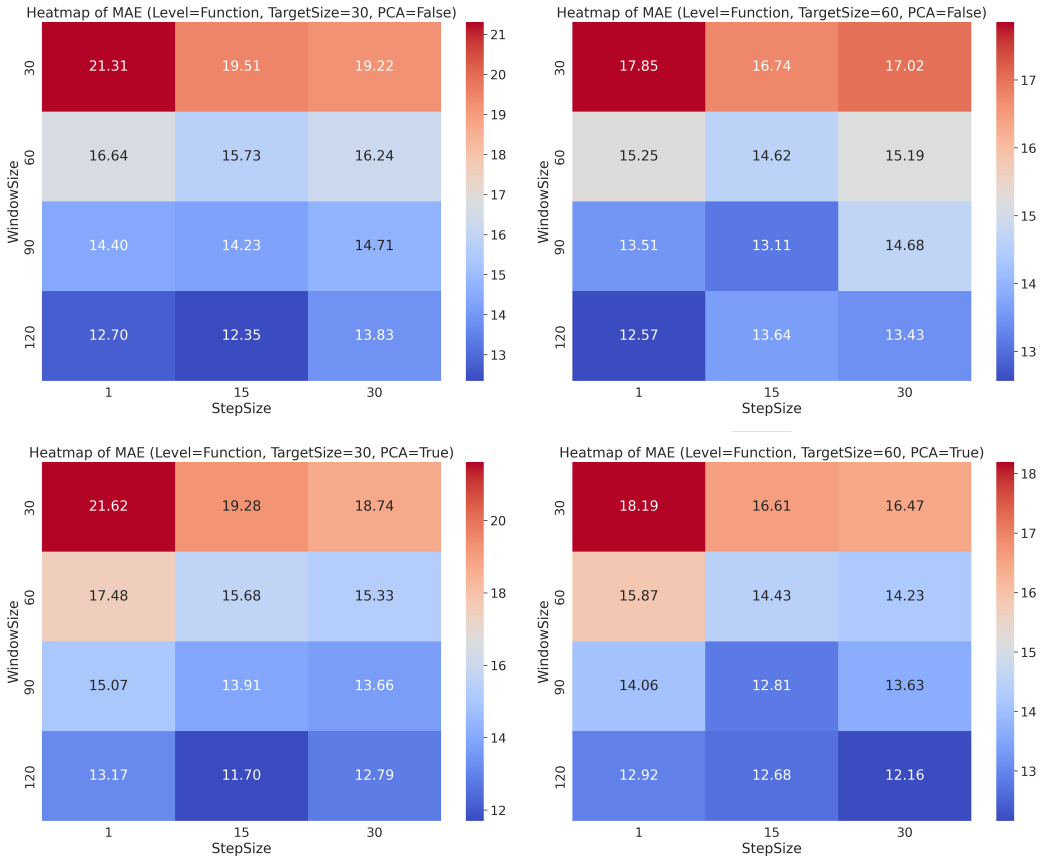


Figure 6: Comparative Heatmaps of Mean Absolute Error for Linear Regression Model: Effects of Window Size, Target Size, and Step Size at Function level With and Without the application of PCA.

and preservation of predictive quality.

At the Function level, the pattern count reduction is consistent with the other levels, and the impact of PCA is again apparent, with a reduction in MAE, emphasizing the effectiveness of PCA in yielding a concise yet powerful set of characteristics for prediction.

Line graphs showing window size, target size, and step size variations offer insight into how different configurations affect the model. A larger window size often results in a lower MAE, but the optimal configuration also depends on the appropriate combination of target size and step size.

The reduction in pattern count at all levels, particularly when PCA is applied, plays a significant role in improving the predictive modeling process. This analysis underscores the importance of careful parameter tuning and the effectiveness of dimensionality reduction techniques in optimizing serverless computing predictive models.

4.6. Model Performance Evaluation

4.6.1. Learning Curves

Figure 10 presents the learning curves of the LR model's performance at each analysis level. These curves are plotted to compare the training and testing MAE, R^2 score, and MSE against the number of patterns used in the time series data. The time series data for each level has been prepared based on the best-performing settings, which achieved the lowest MAE based on previous findings, thus ensuring an optimized learning process.

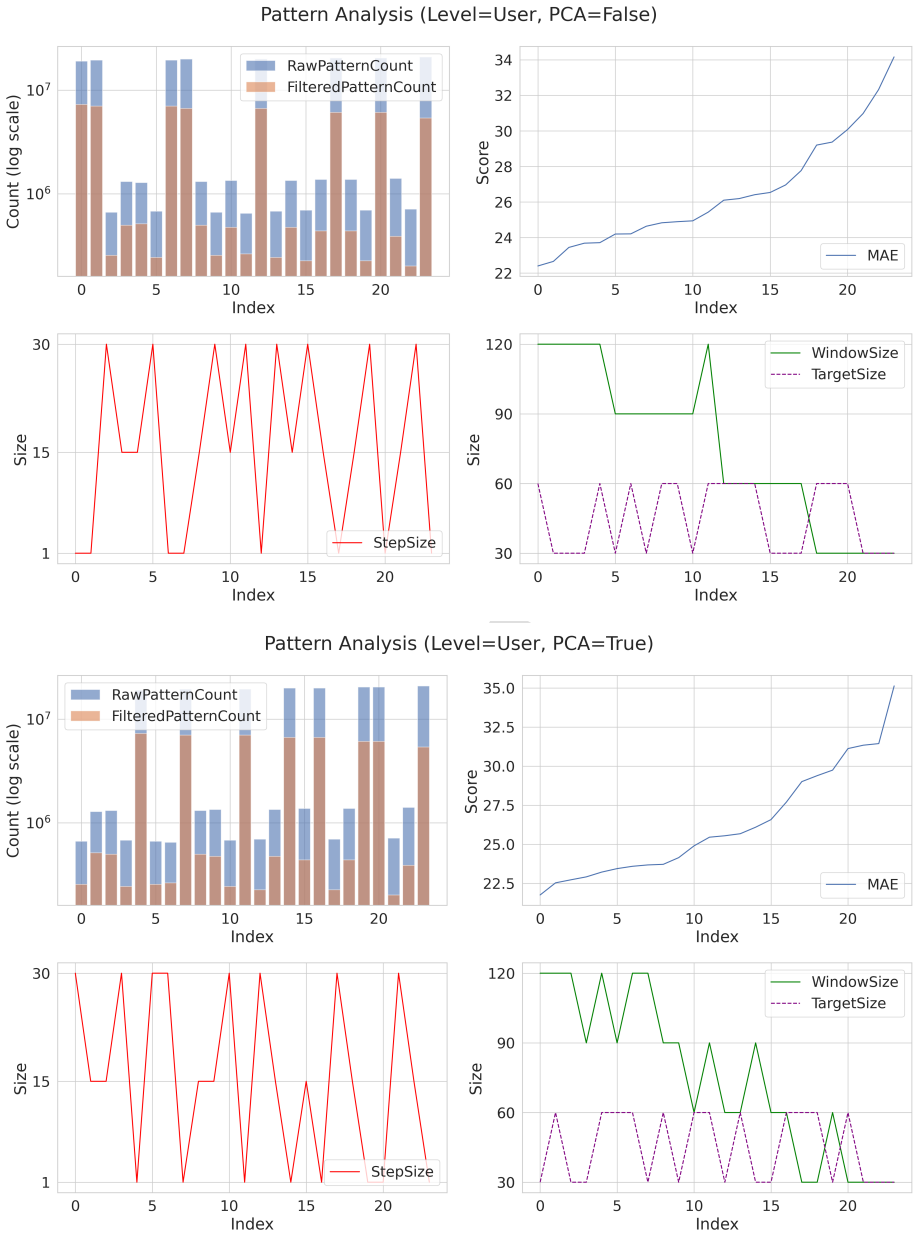


Figure 7: Pattern Analysis using Linear Regression Model: Correlation between Raw and Filtered Pattern Counts, Mean Absolute Error, and Windowing Parameter Variations at the User level with and without the application of PCA.

The learning curves are indicative of the model's ability to learn from a given number of patterns. A decrease in MAE and MSE, along with an increase in the R^2 score for both the training and the test data sets as the number of patterns increases, suggests that the model effectively captures the underlying trends and dynamics of the data. It is

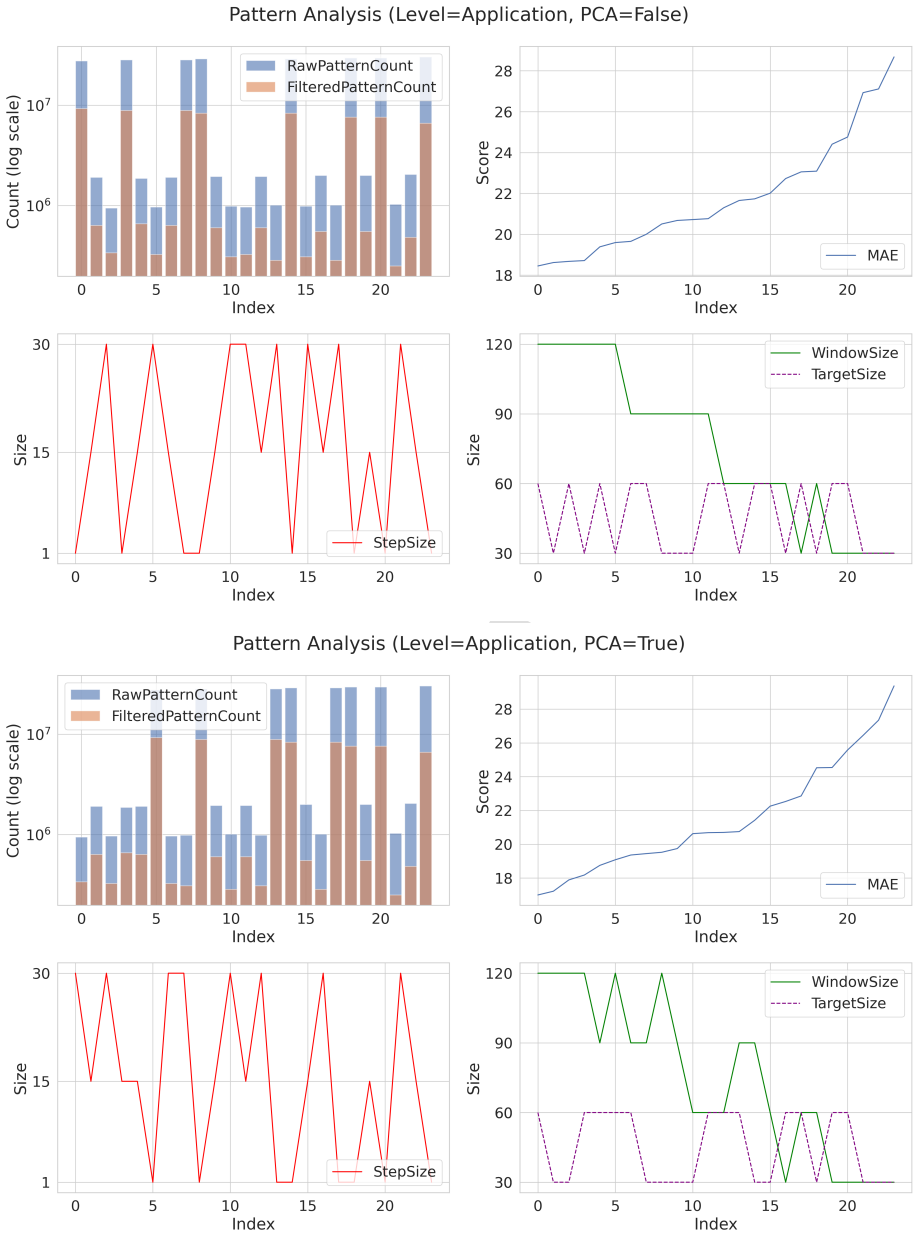


Figure 8: Pattern Analysis using Linear Regression Model: Correlation between Raw and Filtered Pattern Counts, Mean Absolute Error, and Windowing Parameter Variations at the Application level with and without the application of PCA.

evident from the curves that as more data is provided, the model's performance on the test set converges towards its performance on the training set, which is a desirable trait indicating good generalization.

The windowing operation, an essential feature of time series preprocessing, has been employed to improve the

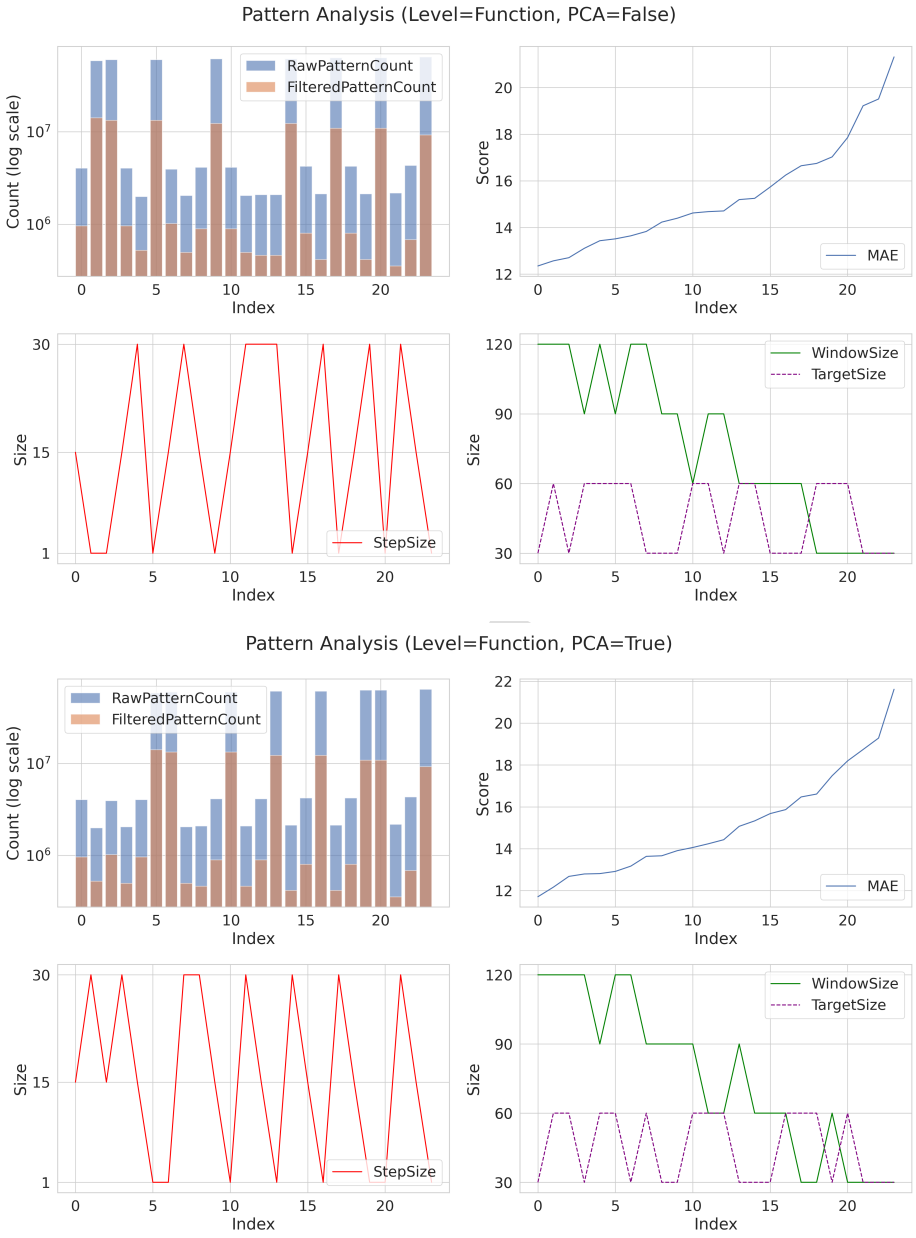


Figure 9: Pattern Analysis using Linear Regression Model: Correlation between Raw and Filtered Pattern Counts, Mean Absolute Error, and Windowing Parameter Variations at the Function level with and without the application of PCA.

learning process. By systematically selecting the best window size, step size, and target size for each level, the model is better able to capture the temporal dependencies and nuances within the data. This operation enhances the learning capability of the model by providing it with a structured format of input data, which is particularly important for the

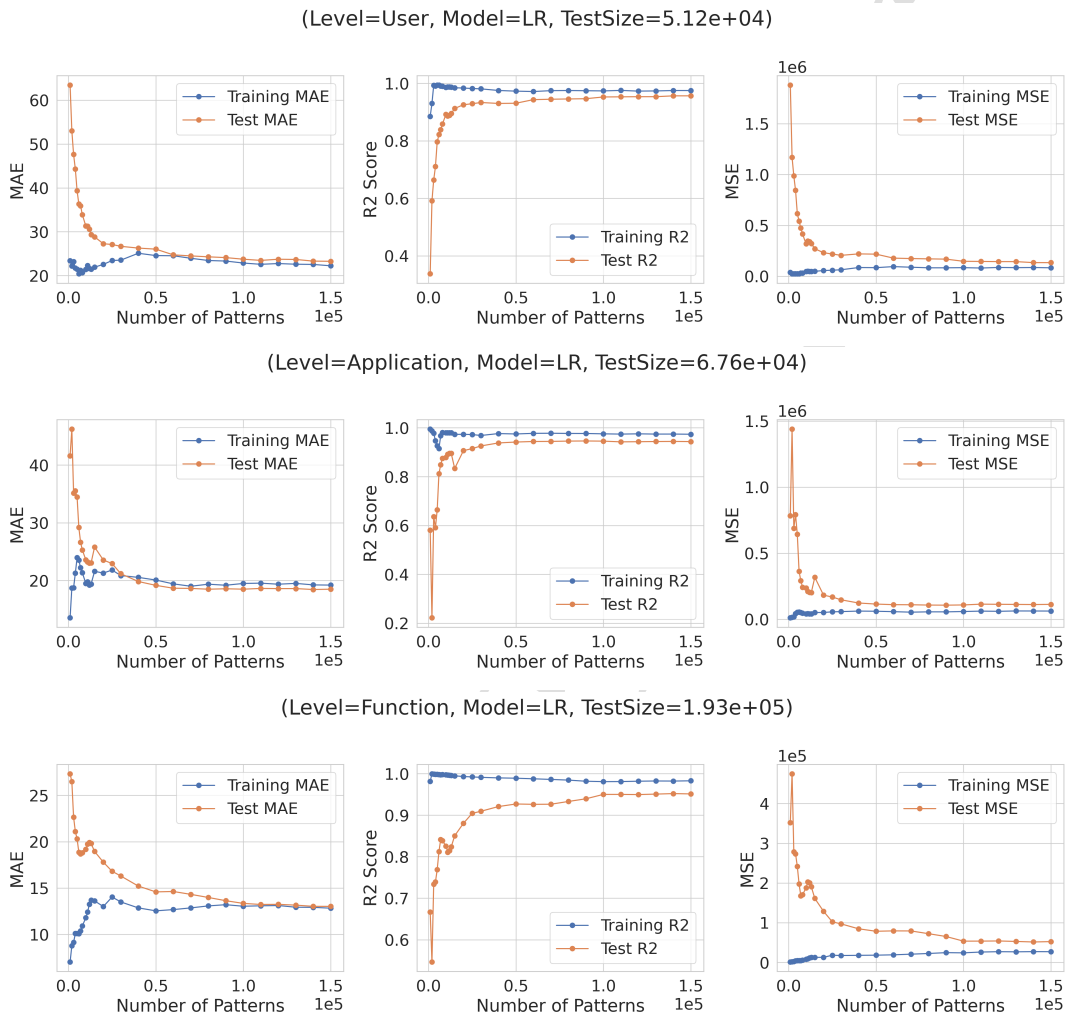


Figure 10: Learning Curves of Linear Regression Model Performance at User, Application, and Function Levels: Comparison of Training and Testing MAE, R2 Score, and MSE Against the Number of Patterns.

regression models used in this study.

In conclusion, the learning curves underscore the importance of careful time series data preparation and the positive impact of the windowing operation on the model's learning process. The convergence of training and testing curves with an increasing number of patterns indicates a well-performing model that generalizes well to new data.

4.6.2. Performance Comparison of Regression Models on Test Data

We present a detailed evaluation of various regression models on test data at the three distinct levels. Table 5 displays the comparative performance metrics for each model, with a particular emphasis on the best-performing settings based on the MAE.

At the User level, as shown in Table 5a, the KNN model outperforms the others with the lowest MAE of **19.19**, indicating its superior accuracy in capturing user behavior within the serverless environment. The ET and RF models

Journal Pre-proof

Table 5: Evaluation of Regression Models on Test Data: Comparative Performance Metrics

(a) Level = User

Model	MAE	MSE	R^2
KNN	19.19	124893	0.9599
ET	19.83	102214	0.9663
RF	20.39	100046	0.9668
LR	21.76	118172	0.9615
DT	26.18	164078	0.9455

(b) Level = Application

Model	MAE	MSE	R^2
KNN	15.02	105949	0.9472
ET	15.97	104460	0.9477
RF	16.20	105512	0.9471
LR	17.00	107271	0.9466
DT	21.88	188880	0.9038

(c) Level = Function

Model	MAE	MSE	R^2
ET	9.71	41832	0.9615
KNN	9.89	53004	0.9519
RF	10.07	44069	0.9596
LR	11.70	48005	0.9558
DT	12.87	79606	0.9268

Table 6: Evaluation of Regressor Performance: Mean Score Analysis of Key Metrics Over Consecutive Days

(a) Level = User

Model	MAE	MSE	R^2
KNN	21.53	206950	0.9403
ET	21.86	205178	0.9408
LR	22.57	196447	0.9428
RF	23.21	222022	0.9361
DT	29.94	330151	0.9045

(b) Level = Application

Model	MAE	MSE	R^2
KNN	17.58	146096	0.9383
ET	18.13	142008	0.9401
LR	18.59	137825	0.9412
RF	19.27	156298	0.9343
DT	25.47	265250	0.8877

(c) Level = Function

Model	MAE	MSE	R^2
ET	11.99	85568	0.9401
KNN	12.07	87829	0.9386
RF	12.57	90116	0.9369
LR	12.85	82413	0.9418
DT	15.86	131258	0.9080

also show commendable performance, with only slight differences in MAE, MSE, and R^2, suggesting their robustness in handling user-level data. The LR model, while not outperforming the ensemble methods, still maintains a competitive R^2 score, highlighting its effectiveness as a simpler alternative. The DT model exhibits the highest MAE and MSE, indicating a relative underperformance in this context.

Moving to the application level, as detailed in Table 5b, we observe a similar pattern with KNN achieving the lowest MAE of **15.02**. The ET and RF models closely follow, with marginal differences in MAE and MSE values but comparable R^2 scores. The LR model, despite its simplicity, presents a reasonable MAE and R^2, underscoring its utility in application-level predictions. The DT model, however, shows a notable decrease in performance, reflected by the highest MAE and the lowest R^2 score among the evaluated models.

At the Function level, Table 5c indicates that the ET model secures the best MAE of **9.71**, closely followed by the KNN and RF models. These models exhibit strong predictive capabilities, as evidenced by their R^2 scores exceeding **0.95**. The LR model, while slightly lagging in MAE, maintains a R^2 score above **0.95**, suggesting its adequacy for function-level prediction tasks. The DT model shows the largest discrepancy in MAE and the lowest R^2 score, implying a less precise fit to the function-level data compared to its counterparts.

For each level, the time series data is meticulously prepared based on the configuration that achieved the lowest MAE in the comparative analysis of the window sliding parameters. This approach ensures that the models are evaluated on data that is optimized for their learning algorithms, providing a fair and rigorous assessment of their predictive performance.

The evaluation of regression models on the test data demonstrates the varying effectiveness of different modeling approaches at the user, application, and function levels. The use of best-performing settings for data preparation has proven to be a decisive factor in enhancing model accuracy, as indicated by the MAE across all levels. The results provide valuable information for selecting appropriate models for serverless computing environments based on the specific requirements of each analytical level.

4.6.3. Evaluation of Regression Models Over Time

Table 6 provides an aggregated evaluation of the performance of different regressor models, analyzing the mean score of key metrics over consecutive days. This longitudinal analysis assesses the stability and reliability of the models at each analysis level.

At the user level, Table 6a shows that the KNN model achieves the lowest mean MAE, indicating its strength to consistently predict user behavior with minimal deviation. The ET and LR models closely follow, with slightly higher

Journal Pre-proof

MAE values but better MSE performance, suggesting their efficiency in minimizing error across predictions. The RF model, while exhibiting a higher MAE and MSE, still maintains a satisfactory R^2 score. The DT model, with the highest MAE and MSE, shows the most significant variation in the predictions over time.

In the context of the application level, as depicted in Table 6b, the KNN model again presents the lowest mean MAE, endorsing its robustness in application-level predictions. The ET model shows comparable performance with a slightly higher MAE but a lower MSE, while the LR model scores the best in terms of R^2, indicating a strong correlation with the observed data. The RF model records a modest increase in mean MAE and MSE, and the DT model ranks last with the highest mean errors scores, pointing to less consistency in its predictions.

For the function level, detailed in Table 6c, the ET model achieves the lowest mean MAE, reinforcing its effectiveness in function-level forecasting. KNN and RF models also perform well, maintaining mean MAE scores within a close range. The LR model, despite a slightly higher MAE, attains the best R^2 score, suggesting that its predictive accuracy is quite reliable. The DT model, as observed at other levels, has the highest mean MAE and MSE, indicating a broader variability in its daily predictions.

This comparative performance analysis over consecutive days highlights the importance of model selection based on consistent performance metrics. Although some models excel in certain metrics, a comprehensive view of all scores is crucial for selecting a model that offers reliability and consistency in a serverless computing environment.

4.6.4. Stability Analysis of Regression Models

Figure 11 illustrates the daily performance trends of various regression models over a 12 day period, at the user, application, and function levels. This analysis provides insights into the consistency and variability of model predictions over time, highlighting the reliability of each regression approach in a dynamic serverless computing environment.

The MAE, R^2 score, and MSE for each type of model (ET, DT, RF, KNN, and LR) are plotted daily. Trends show the extent to which each model is able to maintain consistency of performance on successive days. A stable MAE and a consistently high R^2 score are desirable, indicating that the model's predictions are accurate and reliable over time. The MSE gives a sense of the prediction error's variance; lower values denote more precise predictions. For each level, the stability of the models is crucial for ensuring accurate predictions of serverless function invocations, which are integral to resource allocation and management within serverless architectures.

The comparative analysis of MAE, R^2, and MSE reveals the daily performance stability of the regressors, informing the selection of the most robust model for each level of analysis. It is evident that some models exhibit greater variability in their performance metrics, while others maintain a more consistent trend. This detailed daily performance stability analysis is instrumental in understanding the temporal dynamics of the model predictions. The findings of this analysis are critical for determining the drop performance threshold and, accordingly, inform the redevelopment of the model with the recent data for more reliable and stable predictive models in serverless computing environments.

Figure 12 presents a boxplot summary of the daily performance effectiveness of various regressors and metrics over 12 days for each level. Boxplots are utilized to represent the distribution of the performance metrics, providing insight into the median, interquartile range, and outliers of each regressor's performance over the observed period. They enable the assessment of model stability and reliability for consistent predictive performance across applications. The central line of each box represents the median value, while the top and bottom edges indicate the $75th$ and $25th$ percentiles, respectively. Outliers are depicted as individual points beyond the whiskers of the boxplots.

For each level, the boxplots convey the variability and central tendency in MAE, R^2, and MSE for each regressor. The distribution width signifies the stability of the model's performance. The narrower boxes suggest consistent performance, while the wider ones indicate variability over days. The MAE and MSE boxplots help to understand the magnitude and variance of the error, while the R^2 boxplots indicate the consistency of the predictive accuracy of the model. The comparative size of the boxes and the position of the median line provide an immediate visual cue about the model's performance over the examined days. However, we can observe that the LR regressor has the most stable performance across all levels of analysis, with the smallest spread interquartile range in its boxplot. The KNN, RF, and ET regressors also demonstrate relatively stable performance, albeit not as much as the LR regressor. On the other hand, the DT regressor has a larger spread, indicating more variability in its daily performance.

These boxplot summaries are instrumental in evaluating the daily performance efficacy of regressors. They enable a quick assessment of which models are more stable and reliable over time, which is paramount for tasks requiring consistent predictive performance in a serverless computing environment. They provide a comprehensive overview

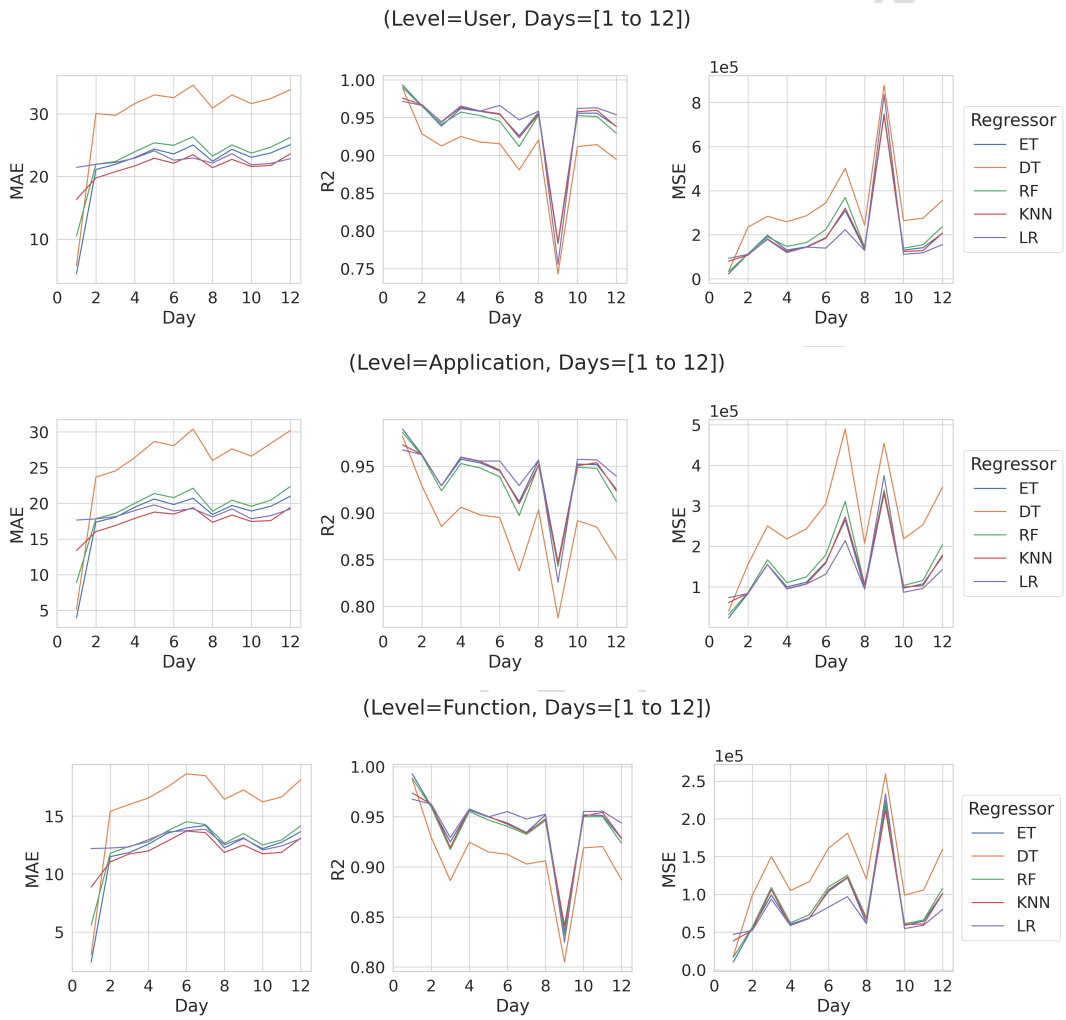


Figure 11: Daily Performance Trends of Various Regressors at User, Application, and Function Levels: Comparative Analysis of MAE, R2 Score, and MSE Over a 12-Day Period.

of the performance variations and stability of different regression models over consecutive days. This graphical summary is essential for selecting the most appropriate model based on the specific needs of user, application, and function levels within serverless architectures.

In our analysis, the LR model exhibited the most stable performance across all levels, characterized by the smallest spread in the interquartile range of its boxplot, suggesting a consistent predictive accuracy. This stability contrasts with models such as DT, which showed high variability in daily performance, indicating a potential for overfitting and sensitivity to nuances in invocation pattern at each level. While the KNN, RF, and ET regressors showed relatively stable performance, they did not achieve the consistency of the LR model. However, it is essential to note that the simplicity and interpretability of LR come with limitations. The assumption of a linear relationship between input and output may only partially capture the complex, non-linear function invocation patterns in serverless computing environments, potentially leading to underfitting. On the other hand, models like RF and ET, despite their computational intensity

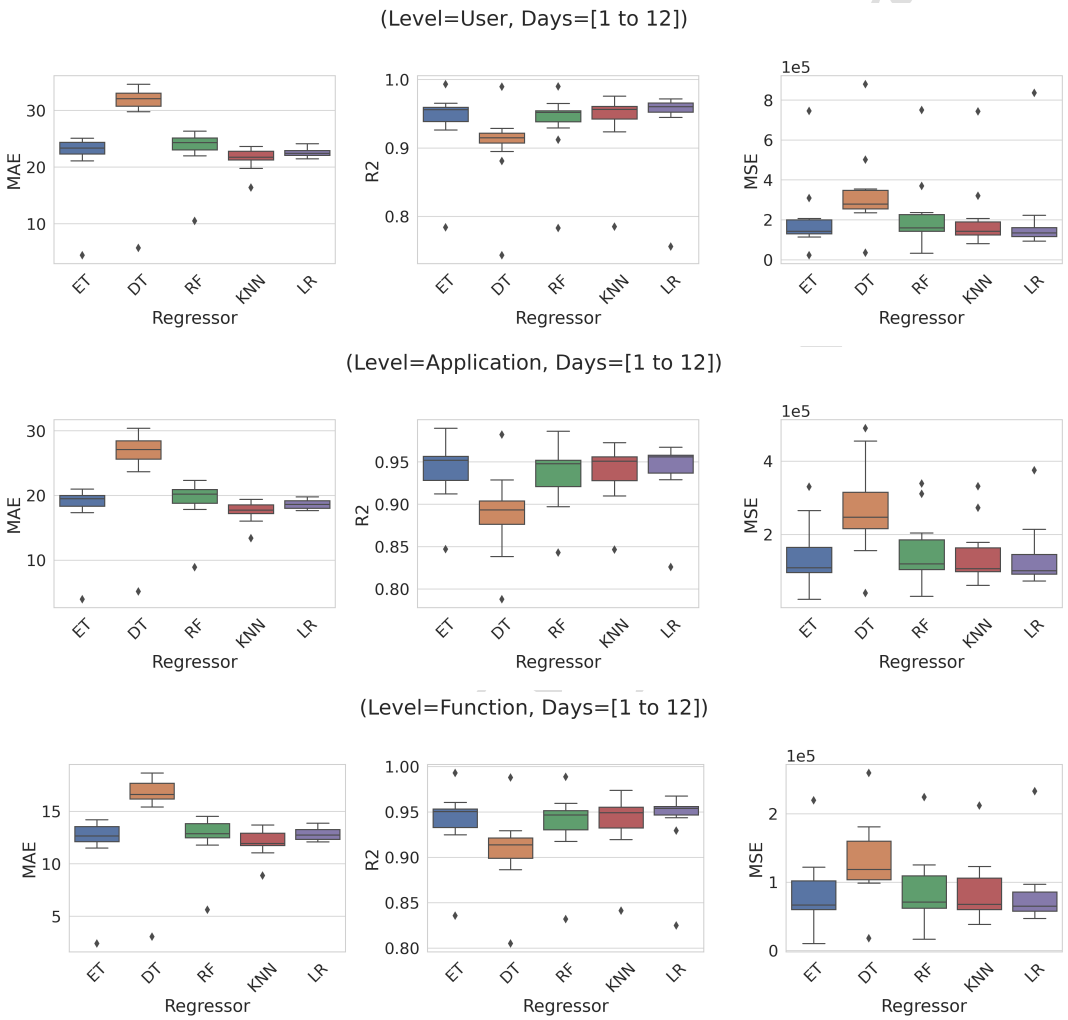


Figure 12: Boxplot Summary of Daily Performance Efficacy of Various Regressors Over 12 Days: Comparing MAE, R2 Score, and MSE Across User, Application, and Function Levels.

as ensemble models, may offer improved predictive capabilities through their ability to model non-linear interactions and complex relationships within the data. Nevertheless, this analysis underscores the importance of considering model-specific biases and limitations, balancing the need for predictive accuracy with computational practicality.

By synthesizing these insights, the study contributes to a deeper understanding of the dynamics at play in serverless computing environments. It offers a set of empirically grounded recommendations for practitioners looking to implement predictive models in such settings, emphasizing the importance of customizing the model configuration to fit the specific needs and constraints of the environment. Furthermore, the findings lay a foundation for future research, suggesting areas for further exploration, such as the development of adaptive models that can dynamically adjust their configuration in response to changing patterns of usage or the integration of additional data sources to enrich the predictive capability of the models. As serverless computing continues to evolve, the insights derived from this comprehensive analysis will play a crucial role in guiding the development of more sophisticated, efficient, and

Journal Pre-proof

accurate predictive models. By continually refining these models, we can better anticipate and respond to the demands of serverless environments, ultimately leading to more robust, scalable, and user-responsive computing solutions.

5. Practical Implications and Future Directions

5.1. Implications for Practice

The findings of this study have several practical implications that can significantly impact deployment strategies in serverless computing environments. By providing a comprehensive multi-level analysis and demonstrating the effectiveness of various predictive modeling techniques, our research offers actionable insights for optimizing serverless architectures.

- **Enhanced Resource Allocation:** The multi-output regression models tailored for user, application, and function levels allow for more precise resource allocation strategies based on the higher precision ability. Practitioners can utilize these models to predict demand and optimize resource provisioning, leading to cost savings and improved performance. For example, a cloud service provider could employ these models to dynamically adjust the computational resources in anticipation of traffic spikes for cloud-based applications, ensuring smooth user experiences during high-demand periods without incurring unnecessary costs during off-peak times.

- **Data Preprocessing Optimization:** The exploration of windowing techniques and dimensionality reduction through PCA highlights the importance of data pre-processing in predictive modeling. Practitioners can apply these techniques to streamline data processing pipelines, enhance model accuracy, and efficiently handle large datasets. A case in point is a streaming service analyzing viewership patterns to predict peak viewing times. By applying dimensionality reduction, the service can process vast amounts of data more efficiently, enabling quicker adjustments to content delivery networks to maintain optimal streaming quality.

- **Model Selection and Configuration:** Our comparative analysis framework provides a blueprint for selecting and configuring the most effective predictive models. By adopting this framework, practitioners can better understand the trade-offs between different models and configurations, leading to more informed decisions in the deployment of predictive systems. An Internet of Things (IoT) platform could leverage this framework to choose the best model for forecasting device data transmission rates, ensuring adequate bandwidth, and reducing the likelihood of network congestion.

- **Longitudinal Monitoring and Adjustment:** The study's focus on the temporal stability of the models encourages practitioners to adopt a longitudinal approach to performance monitoring. This ongoing assessment ensures that predictive systems remain accurate and reliable over time, adapting to changes in workload patterns.

By integrating these findings into their operational strategies, serverless computing practitioners can significantly enhance their systems' predictability, efficiency, and overall performance.

5.2. Future Research

The results of this study open several avenues for future research, with the aim of further refine predictive function invocation methods and extend the applicability of our findings.

- **Advanced Predictive Models:** Future research could explore developing and integrating more advanced predictive models, including multi-level learning strategies, deep learning, and advanced ensemble methods. These models might capture more complex patterns and interactions within serverless workloads, potentially improving predictive accuracy.

- **Broader Dataset Application:** While this study utilized the Azure Functions data set version 2019, subsequent research should consider applying the developed methodologies to a more comprehensive range of data sets from different cloud providers. This broader application will help validate the generalizability of the models and techniques.

Journal Pre-proof

- **Real-time Prediction Capabilities:** Investigating the feasibility and effectiveness of real-time predictive modeling in serverless environments represents a significant research opportunity. This exploration includes evaluating the challenges and benefits of implementing real-time prediction systems and their impact on operational efficiency.
- **Cost-Benefit Analysis:** Future studies should include a detailed cost-benefit analysis of the implementation of predictive function invocation methods. This analysis would help quantify the economic impacts and provide a more comprehensive understanding of the benefits and drawbacks of different approaches.
- **Enhanced Dimensionality Reduction Techniques:** Expanding on the PCA findings, future research might explore other dimensionality reduction techniques and their efficacy in further optimizing model performance and computational efficiency.

By addressing these areas, future research can continue to advance the field of serverless computing, offering increasingly sophisticated tools and methodologies for optimizing cloud resources and improving service delivery.

6. Conclusion

This study has presented a comprehensive exploration into enhancing the predictability and efficiency of function invocations in serverless computing environments. Through a systematic approach employing multi-output regression models, windowing techniques, and PCA for dimensionality reduction, we have provided valuable insights and methodologies that push forward the capabilities of serverless computing.

1. Our multi-level predictive modeling has demonstrated significant potential in understanding and predicting function invocation patterns across user, application, and function levels. This granular approach is pivotal for fine-tuning resource allocation and improving operational efficiency in cloud environments.
2. The detailed exploration of windowing techniques and the strategic application of PCA have revealed the importance of optimizing data preprocessing and feature extraction in predictive modeling. Our findings emphasize the balance between maintaining data integrity and computational efficiency, a critical consideration in large-scale data environments.
3. The development of a comparative analysis framework and the utilization of a real-world cloud workload trace have allowed for a thorough evaluation of model performances. This framework is instrumental in identifying optimal configurations and ensuring that the predictive models are not only theoretically sound but also practically viable.
4. Furthermore, the assessment of temporal stability and performance variations of the models over consecutive days contributes to the reliability and robustness of predictive systems in serverless computing, addressing a significant challenge in the field.
5. Lastly, our research has outlined several pathways for future work, encouraging continued advancements in the predictive modeling of serverless computing workloads.

The contributions of this study are intended to serve as a foundation for future research and practical applications in serverless computing. We advocate for continued exploration and innovation in this domain, as the accurate prediction of function invocations is paramount in optimizing cloud resources and enhancing service delivery. As serverless architectures evolve, so too must the methodologies and tools designed to support them, ensuring that they remain efficient, cost-effective, and responsive to the needs of diverse applications.

References

[1] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, et al., Serverless computing: Current trends and open problems, Research advances in cloud computing (2017) 1–20doi:10.1007/978-981-10-5026-8_1.

[2] B. Wang, A. Ali-Eldin, P. Shenoy, Lass: Running latency sensitive serverless computations at the edge, in: Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing, HPDC '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 239–251. doi:10.1145/3431379.3460646.
URL https://doi.org/10.1145/3431379.3460646

Journal Pre-proof

[3] D. Ustiugov, T. Amariucai, B. Grot, Analyzing tail latency in serverless clouds with stellar, in: 2021 IEEE International Symposium on Workload Characterization (IISWC), 2021, pp. 51–62. doi:10.1109/IISWC53511.2021.00016.

[4] S. McAleese, J. C. McLaughlin, F. Detyna, A. Murashev, M. Yilmaz, P. M. Clarke, Serverless software engineering – and how to get there., in: M. Yilmaz, P. Clarke, R. Messnarz, B. Wöran (Eds.), Systems, Software and Services Process Improvement, Springer International Publishing, Cham, 2022, pp. 75–90.

[5] G. Casale, M. Artač, W. v. d. Heuvel, A. v. Hoorn, P. Jakovits, F. Leymann, M. Long, V. K. Papanikolaou, D. Presenza, A. Russo, S. N. Srirama, D. A. Tamburri, M. Wurster, L. Zhu, Radon: rational decomposition and orchestration for serverless computing, SICS Software-Intensive Cyber-Physical Systems 35 (2020) 77–87. doi:10.1007/s00450-019-00413-w.

[6] S. Lee, D. Yoon, S. Yeo, S. Oh, Mitigating cold start problem in serverless computing with function fusion, Sensors 21 (24) (2021). doi:10.3390/s21248416.
URL https://www.mdpi.com/1424-8220/21/24/8416

[7] I. Müller, R. Marroquín, G. Alonso, Lambada: Interactive data analytics on cold data using serverless cloud infrastructure, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, SIGMOD '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 115–130. doi:10.1145/3318464.3389758.
URL https://doi.org/10.1145/3318464.3389758

[8] Y. Yuan, X. Shi, Z. Lei, X. Wang, X. Zhao, Smpi: Scalable serverless mpi computing, in: 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC), 2022, pp. 275–282. doi:10.1109/IPCCC55026.2022.9894339.

[9] A. Luckow, S. Jha, Performance characterization and modeling of serverless and hpc streaming applications, in: 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 5688–5696. doi:10.1109/BigData47090.2019.9006530.

[10] R. B. Roy, T. Patel, D. Tiwari, Icebreaker: Warming serverless functions better with heterogeneity, in: Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 753–767. doi:10.1145/3503222.3507750.
URL https://doi.org/10.1145/3503222.3507750

[11] P. Castro, V. Ishakian, V. Muthusamy, A. Slominski, The rise of serverless computing, Commun. ACM 62 (12) (2019) 44–54. doi:10.1145/3368454.
URL https://doi.org/10.1145/3368454

[12] A. Alhindi, K. Djemame, F. B. Heravan, On the power consumption of serverless functions: An evaluation of openfaas, in: 2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC), 2022, pp. 366–371. doi:10.1109/UCC56403.2022.00064.

[13] A. Pérez, G. Moltó, M. Caballer, A. Calatrava, Serverless computing for container-based architectures, Future Generation Computer Systems 83 (2018) 50–59. doi:https://doi.org/10.1016/j.future.2018.01.022.
URL https://www.sciencedirect.com/science/article/pii/S0167739X17316485

[14] J. Carreira, S. Kohli, R. Bruno, P. Fonseca, From warm to hot starts: Leveraging runtimes for the serverless era, in: Proceedings of the Workshop on Hot Topics in Operating Systems, HotOS '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 58–64. doi:10.1145/3458336.3465305.
URL https://doi.org/10.1145/3458336.3465305

[15] J. Spillner, C. Mateos, D. A. Monge, Faaster, better, cheaper: The prospect of serverless scientific computing and hpc, in: E. Mocskos, S. Nesmachnow (Eds.), High Performance Computing, Springer International Publishing, Cham, 2018, pp. 154–168.

[16] L. Schuler, S. Jamil, N. Kühl, Ai-based resource allocation: Reinforcement learning for adaptive auto-scaling in serverless environments, in: 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2021, pp. 804–811. doi:10.1109/CCGrid51090.2021.00098.

[17] R. Tolosana-Calasanz, G. G. Castañé, J. Á. Bañares, O. Rana, Modelling serverless function behaviours, in: K. Tserpes, J. Altmann, J. Á. Bañares, O. Agmon Ben-Yehuda, K. Djemame, V. Stankovski, B. Tuffin (Eds.), Economics of Grids, Clouds, Systems, and Services, Springer International Publishing, Cham, 2021, pp. 109–122.

[18] D. Ustiugov, P. Petrov, M. Kogias, E. Bugnion, B. Grot, Benchmarking, analysis, and optimization of serverless function snapshots, in: Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 559–572. doi:10.1145/3445814.3446714.
URL https://doi.org/10.1145/3445814.3446714

[19] P. Raith, faas-sim: a trace-driven simulation framework for serverless edge computing platforms, Software Practice and Experience (2023). doi:10.1002/spe.3277.

[20] S. Arbat, V. Jayakumar, J. Lee, W. Wang, I. Kim, Wasserstein adversarial transformer for cloud workload prediction, Proceedings of the Aaai Conference on Artificial Intelligence 36 (2022) 12433–12439. doi:10.1609/aaai.v36i11.21509.

[21] X. Wei, F. Lu, T. Wang, J. Gu, Y. Yang, R. Chen, H. Chen, No provisioned concurrency: Fast RDMA-codesigned remote fork for serverless computing, in: 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23), USENIX Association, Boston, MA, 2023, pp. 497–517.
URL https://www.usenix.org/conference/osdi23/presentation/wei-rdma

[22] E. van Eyk, J. Scheuner, S. Eismann, C. L. Abad, A. Iosup, Beyond microbenchmarks: The spec-rg vision for a comprehensive serverless benchmark, in: Companion of the ACM/SPEC International Conference on Performance Engineering, ICPE '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 26–31. doi:10.1145/3375555.3384381.
URL https://doi.org/10.1145/3375555.3384381

[23] S. Ilager, K. Ramamohanarao, R. Buyya, Etas: Energy and thermal-aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation, Concurrency and Computation: Practice and Experience 31 (17) (2019) e5221.

[24] J. A. Adeleke, D. Moodley, G. Rens, A. O. Adewumi, Integrating statistical machine learning in a semantic sensor web for proactive monitoring and control, Sensors 17 (4) (2017) 807.

[25] B. L. Dalmazo, J. P. Vilela, M. Curado, Online traffic prediction in the cloud: a dynamic window approach, in: 2014 International Conference on Future Internet of Things and Cloud, IEEE, 2014, pp. 9–14. doi:10.1109/ficloud.2014.12.

Journal Pre-proof

[26] H. Mehdi, Z. Pooranian, P. G. Vinueza Naranjo, Cloud traffic prediction based on fuzzy arima model with low dependence on historical data, Transactions on Emerging Telecommunications Technologies 33 (3) (2022) e3731.

[27] J. R. Beattie, F. W. Esmonde-White, Exploration of principal component analysis: deriving principal component analysis visually using spectra, Applied Spectroscopy 75 (4) (2021) 361–375.

[28] M. Ghorbani, E. K. Chong, Stock price prediction using principal components, PloS one 15 (3) (2020) e0230124.

[29] W. Liu, Z. Sun, J. Chen, C. Jing, et al., Raman spectroscopy in colorectal cancer diagnostics: Comparison of pca-lda and pls-da models, Journal of Spectroscopy 2016 (2016).

[30] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, R. Buyya, ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, Software: Practice and Experience 47 (9) (2017) 1275–1296.

[31] J. Pang, D. Liu, Y. Peng, X. Peng, Collective anomalies detection for sensing series of spacecraft telemetry with the fusion of probability prediction and markov chain model, Sensors 19 (3) (2019) 722.

[32] L. Zhang, Y. Zhang, P. Jamshidi, L. Xu, C. Pahl, Service workload patterns for qos-driven cloud resource management, Journal of Cloud Computing 4 (2015) 1–21.

[33] M. Steinbach, A. Jindal, M. Chadha, M. Gerndt, S. Benedict, Tppfaas: modeling serverless functions invocations via temporal point processes, Ieee Access 10 (2022) 9059–9084. doi:10.1109/access.2022.3144078.

[34] Z. liu, X. Xu, Studying the impact of health education on student knowledge and behavior through big data and cloud computing, Scientific Programming 2022 (2022) 1–11. doi:10.1155/2022/4160821.

[35] L. Zhang, B. Zhang, C. Pahl, L. Xu, Z. Zhu, Personalized quality prediction for dynamic service management based on invocation patterns, in: S. Basu, C. Pautasso, L. Zhang, X. Fu (Eds.), Service-Oriented Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 84–98.

[36] A. Mampage, S. Karunasekera, R. Buyya, A holistic view on resource management in serverless computing environments: taxonomy and future directions, Acm Computing Surveys 54 (2022) 1–36. doi:10.1145/3510412.

[37] N. Mahmoudi, H. Khazaei, Performance modeling of serverless computing platforms, Ieee Transactions on Cloud Computing 10 (2022) 2834–2847. doi:10.1109/tcc.2020.3033373.

[38] H. Hassan, S. Barakat, Q. Sarhan, Survey on serverless computing, Journal of Cloud Computing Advances Systems and Applications 10 (2021). doi:10.1186/s13677-021-00253-7.

[39] K. R. Rajput, C. D. Kulkarni, B. Cho, W. Wang, I. K. Kim, Edgefaasbench: Benchmarking edge devices using serverless computing, in: 2022 IEEE International Conference on Edge Computing and Communications (EDGE), IEEE, 2022, pp. 93–103. doi:10.1109/edge55608.2022.00024.

[40] T. Yu, Q. Liu, D. Du, Y. Xia, B. Zang, Z. Lu, P. Yang, C. Qin, H. Chen, Characterizing serverless platforms with serverlessbench, in: Proceedings of the 11th ACM Symposium on Cloud Computing, 2020, pp. 30–44. doi:10.1145/3419111.3421280.

[41] Z. Li, Y. Liu, L. Guo, Q. Chen, J. Cheng, W. Zheng, M. Guo, Faasflow: Enable efficient workflow execution for function-as-a-service, in: Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 2022, pp. 782–796. doi:10.1145/3503222.3507717.

[42] Z. Li, L. Guo, J. Cheng, Q. Chen, B. He, M. Guo, The serverless computing survey: A technical primer for design architecture, ACM Computing Surveys (CSUR) 54 (10s) (2022) 1–34.

[43] D. Senthil, G. Suseendran, Efficient time series data classification using sliding window technique based improved association rule mining with enhanced support vector machine, International Journal of Engineering & Technology 7 (3.3) (2018) 218. doi:10.14419/ijet.v7i2.33.13890.

[44] M. Shahrad, R. Fonseca, I. Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, R. Bianchini, Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider, in: 2020 USENIX Annual Technical Conference (USENIX ATC 20), USENIX Association, 2020, pp. 205–218.
URL `https://www.usenix.org/conference/atc20/presentation/shahrad`

[45] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, R. Bianchini, Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms, in: Proceedings of the 26th Symposium on Operating Systems Principles, SOSP '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 153–167. doi:10.1145/3132747.3132772.
URL `https://doi.org/10.1145/3132747.3132772`

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[47] S. Manhas, R. Vashisht, P. S. Sandhu, N. Neeru, Reusability evaluation model for procedurebased software systems, International Journal of Computer and Electrical Engineering 2 (6) (2010) 1107.

[48] V. V. Nguyen, B. T. Pham, B. T. Vu, I. Prakash, S. Jha, H. Shahabi, A. Shirzadi, D. N. Ba, R. Kumar, J. M. Chatterjee, et al., Hybrid machine learning approaches for landslide susceptibility modeling, Forests 10 (2) (2019) 157.

[49] D. B. Figueiredo Filho, J. A. S. Júnior, E. C. Rocha, What is r2 all about?, Leviathan (São Paulo) (2011) 60–68.

[50] M. Daraghmeh, A. Agarwal, Y. Jararweh, Incorporating data preparation and clustering techniques for workload segmentation in large-scale cloud data centers, in: 2023 Fourth International Conference on Intelligent Data Science Technologies and Applications (IDSTA), IEEE, 2023, pp. 7–14.

[51] M. Daraghmeh, A. Agarwal, Y. Jararweh, An ensemble clustering approach for modeling hidden categorization perspectives for cloud workloads, Cluster Computing (2023) 1–25doi:10.1007/s10586-023-04205-5.

[52] M. Daraghmeh, A. Agarwal, Y. Jararweh, A multilevel learning model for predicting cpu utilization in cloud data centers, in: 2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), IEEE, 2023, pp. 1016–1023.

[53] M. Daraghmeh, A. Agarwal, Y. Jararweh, Regression-based approach for proactive predictive modeling of efficient cloud cost estimation, in: 2023 Tenth International Conference on Software Defined Systems (SDS), IEEE, 2023, pp. 65–72.