Analyzing and Predicting the Power Consumption of a Publish/Subscribe IoT-Broker

Franc Pouhela*, Maryam Arabshahi*, Hans D. Schotten*†

*German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern, Germany
Email: {franc.pouhela; maryam.arabshahi; hans_dieter.schotten}@dfki.de
†University of Kaiserslautern-Landau (RPTU), Germany
Email: {schotten}@rptu.de

***Abstract*—The rapid expansion of the Internet of Things (IoT) has underscored the need for energy-efficient communication protocols, with the Publish/Subscribe model standing out for its efficiency and scalability. This paper presents a comprehensive analysis of the power consumption patterns of a Publish/Subscribe IoT broker, identifying key factors influencing energy consumption. By employing advanced predictive modeling techniques, we developed accurate power consumption models. Our findings highlight the significant impact of message throughput, thread count, and network conditions on power consumption. By comparing a variety of predictive models using different algorithms, we can identify the most suitable model for the task. These predictive models can assist in designing energy-efficient IoT systems, thereby contributing to the development of sustainable IoT infrastructures.**

***Index Terms*—IoT, Power Consumption, Machine Leaning, 6G**

I. Introduction

Considerable effort is directed towards harnessing Artificial Intelligence (AI) across various systems to detect potential anomalies, thereby proactively addressing them and enhancing resilience. This endeavor gains heightened significance in environments characterized by elevated demands for safety, security, scalability, and other critical factors.

In the realm of mobile networks (5G, 6G), there is a notable emphasis on expending network capabilities beyond current limitations. This is achieved by exploring innovative approaches to incorporate AI at various stages of the network infrastructure. The goal is to enhance network orchestration and ensure proper operation by effectively detecting and addressing various anomalies.

An area showing promise under consideration is the concept of a Context Management System (CoMaS) [1]. A CoMaS framework excels in utilizing network context data to gain new insights into network conditions through a variety of AI algorithms and reasoning. These insights are then applied to enhance and streamline network operations.

Figure 2 illustrates the architecture of a CoMaS, which facilitates the collection, processing, and distribution of contextual data to end-users using a Publish/Subscribe messaging model orchestrated by a middleware known as a broker (see Figure 1). This messaging pattern is quite scalable and flexible due to its decoupled nature. The effectiveness of such a system relies heavily on its seamless and efficient acquisition and distribution of contextual data from diverse sources, a challenge compounded by the proliferation of IoT devices. The adoption of Message Queuing Telemetry Transport (MQTT) [2] or Middleware Message Queuing Protocol (MMQP) [3] as the communication protocol for such system would be of good value. MQTT, renowned for its lightweight messaging capabilities, is widely used in both IoT and Machine-to-Machine (M2M) scenarios, currently serving as the predominant protocol for enabling IoT communication.

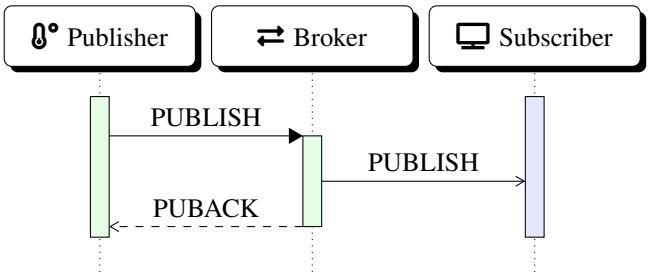

Fig. 1. Publish/Subscribe Pattern

Predicting the power consumption of a system can significantly enhance the detection of potential anomalies in communication scenarios, such as those found in the CoMaS described above. Power efficiency has become an increasingly important concern in the design and operation of communication systems especially in the IoT space. However, our objective surpasses the mere comprehension and accurate prediction of the broker's power consumption; instead, we endeavor to harness this insight as a building stone within our CoMaS to improve anomaly detection, fortify resilience and optimize operational efficiency.

Machine Learning (ML) has the potential to address this challenge by enabling the automated optimization of broker configurations based on predictions. By modeling the power consumption of the broker as a function of various input factors, ML algorithms can learn to identify configurations that result in the lowest power consumption while still meeting performance targets [4]. In addition, reinforcement learning techniques can be used to train agents that can learn to optimize power consumption and performance by taking actions

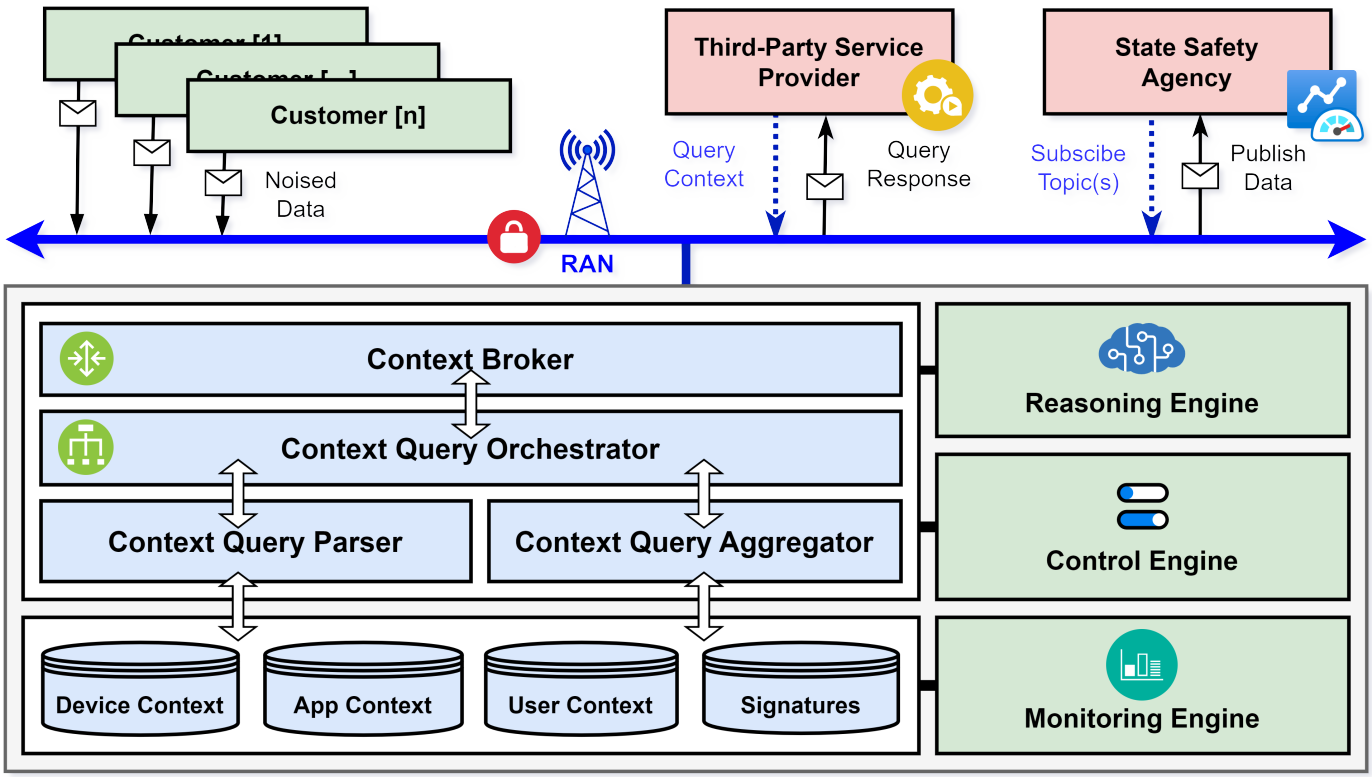

Fig. 2. Context Management System Architecture [1]

such as adjusting hard- and software configurations [5]. The following steps were undertaken in our research:

- Identification of key components of the system that contribute to its power consumption. These include traffic data, processor temperature, memory usage, etc.
- Measurement of the power consumption of the system under a variety of workloads and operating conditions.
- Collection of data on the operating conditions of the system (e.g., processor frequency, memory usage, network activity) while measuring the power consumption. This allowed us to identify relationships between the operating conditions and the power consumption.
- Analysis of the collected data to identify patterns and trends in the power usage of the system using statistical and ML techniques to identify relationships between the operating conditions and the power consumption.
- Development of the power consumption model of the system based on the trends identified in the data.
- Use the model to predict the power consumption of the system under different operating conditions and compare the predictions to actual measurements.
- Comparison on multiple models and fine-tuning the most accurate for better results

The remaining sections of the paper are structured as follows: Section II presents a concise review of related works pertinent to this study. In Section III, we detail our multi-variable model for estimating the energy consumption of a multi-threaded process. Subsequently, Section IV offers an extensive explanation of the data analysis conducted to extract various patterns and trends from the dataset, along with the outcomes of our training and predictions. Finally, Section V summarizes the study and proposes potential avenues for future research.

II. Related Work

In a prior study detailed in [6], we demonstrated the advantages in terms of power efficiency gained by adopting Entity-Component-System (ECS) as the underlying architecture for implementing the IoT broker. The study also provides some insights into the enhancements made to the MQTT protocol, culminating in the development of the MMQP protocol.

There has been a significant amount of research on optimizing the power efficiency of servers and other compute-intensive systems. Many of these approaches focus on Dynamic Voltage and Frequency Scaling (DVFS) as a means of reducing power consumption [7], [8], [9]. DVFS allows the operating frequency and voltage of the processor to be adjusted at runtime in order to meet the performance requirements of the workload while minimizing power consumption. However, these approaches often require hardware support and can be difficult to implement in practice.

Other approaches to improving power efficiency in servers have focused on optimizing the allocation of tasks to threads and cores [10], [11], [12]. By carefully scheduling tasks and minimizing idle time, it is possible to reduce the power consumption of the system without sacrificing performance.

However, these approaches are often specific to the workload and may not generalize well to other scenarios.

In contrast, our approach uses machine learning to predict the power consumption of a multi-threaded server process under different workloads. By using this prediction to optimize the allocation of tasks to threads, we are able to significantly reduce the power consumption of the system while maintaining good performance. To the best of our knowledge, this is the first work that uses machine learning in this way to improve the power efficiency of a multi-threaded server process.

III. Multivariable Energy Model

This section provides critical background information on power profiling and modeling with an emphasis on Intel Central Processing Units (CPUs) used in laptops, desktop computers, and servers. Our study was specifically focused on the power consumption of the CPU. Other components such as the Graphics Processing Unit (GPU) and the Random Access Memory (RAM) were not considered.

A. Hard- and Software Setup

The processor used during our case study is an *11th Gen Intel(R) i7-11800H x64* processor. It has 8 cores and 16 threads, with L1, L2, and L3 cache sizes of 640 KB, 10 MB, and 24 MB, respectively. a Thermal Design Power (TDP) of 45W it has a base frequency of 2.3 GHz and can leverage DVFS to achieve a maximum frequency of 4.5 GHz, and a minimum frequency of 0.8 GHz. The CPU offers a performance boost through hyper-threading. The system has 32 GB of RAM operating at a frequency of 3.2 GHz and runs on the Windows 11 operating system.

1) Measurement Software: The power profiling was made using the Intel® Power Gadget (IPG), which is a software-based power usage monitoring tool for Intel® Core(TM) processors. It provides drivers that can be used in C++ for monitoring and estimating real-time processor package power information in watts using the CPU's energy counters. it has the ability to collect power information on a variety of platforms, including notebooks, desktops, etc.

2) Broker Application: As part of our study, we undertook the development of a custom broker. This last was implemented using C++17 and was carefully crafted using an ECS architecture. ECS is a software architecture design pattern mostly utilized in the development of video games for the representation of in-game objects, which is different from the common Object-Oriented Programming (OOP) technique. Its main function is to separate data from behaviors to promote code reuse and cache-friendliness and enhance performance. To handle I/O operations with utmost efficiency, we integrated the standalone version of Asio [13], a highly performant C++ networking library. Further details on the implementation and performance are available in the cited reference [6].

Our study aimed to explore the capabilities and potential of this implementation approach. Therefore, we conducted thorough evaluations and measurements to assess its performance, scalability, and reliability under different scenarios, providing valuable insights into its suitability for real-world deployment in various contexts within the IoT domain.

B. Model for frequency scaling

The execution time T [sec] of an application depends on the number of threads (n) used and the chosen operational frequency (f), which in turn affects the power drawn P [Watt]. To establish the relationship between energy consumption $E = T \cdot P$ [Joule] and the number of threads, and the operational frequency, we leveraged an energy model that uses n and s as variables. Where s is the scaling factor of f with $s \leqslant 1$. Considering a CPU with a clock frequency of f and a voltage of V. The power consumption of the CPU can be modeled as

$$P = C_{\text{static}} V + C_{\text{dyn}} V^2 f$$

where C_{static} and C_{dyn} are constants that depend on the specific characteristics of the CPU. The first term, $C_{\text{static}}V$, represents the static power consumption, which is the power consumed by the CPU when it is idle or in a low-power state. This term is also proportional to the voltage and is typically much smaller than the dynamic power consumption. The second term in the equation, $C_{\text{dyn}}V^2 f$, represents the dynamic power consumption, which is the power consumed by the CPU when it is actively performing computations. This term is proportional to the clock frequency and the voltage and is typically the dominant contributor to the CPU's power consumption.

Considering the frequency scaling factor $s \leqslant 1$ of DVFS processors, the operational frequency can be expressed as $f = f_{max}/s$, where $f \leqslant f_{max}$ and f_{max} is the highest possible frequency for the processor. In our case $f_{max} = 4.5GHz$.

1) Multi-Variable Equation: A general expression of the energy consumption of a CPU during an execution time t in relation to the frequency scaling factor can be modeled as:

$$E(s) = \int P(s).dt$$

This implies that:

$$E(n, s_v, s_f) = n * K_n * V_{max}^2 * f_{max} * (t_{comp} * s_v^2 * s_f + t_{io} * s_{io})$$

This formula represents the energy consumption E of a system, which depends on several parameters:

- n: Number of threads - s_v: Voltage scale factor - s_f: Frequency scale factor - K_n: Coefficient - V_{max}: Maximum voltage - f_{max}: Maximum frequency - t_{comp}: Time for computation - s_{io}: Scalability factor for input/output operations - t_{io}: Time for input/output operations

This formula describes an approximated model of the energy consumption influenced by the number of operations n, the scalability factors s_v and s_f, the maximum voltage V_{max}, the maximum frequency f_{max}, and the time taken for computation t_{comp} and input/output operations t_{io}.

IV. Methodology and Results

Regression methods in machine learning are a fundamental set of techniques used to model the relationship between a dependent variable and one or more independent variables. The primary goal of regression analysis is to predict the value of the dependent variable based on the values of the independent variables. These methods encompass a wide range of algorithms, from simple linear regression to more complex models like polynomial regression, decision trees, support vector machines, and neural networks. Regression models are widely employed in various fields, including finance, economics, healthcare, and engineering, to forecast trends, understand relationships between variables, and make informed decisions based on data-driven insights.

A. Dataset

The data generated for analysis is depicted in Figure 3. The dataset comprises 11 features, with 8 of them having a data type of "int64" and the remaining 3 featuring a data type of "float64". In total, the dataset consists of 9334 data-points.

	msg_sent	msg_read	bytes_sent	bytes_read	cpu_power	cpu_freq	cpu_temp	cpu_usage
0	0	0	0	0	4.28058	1.1	48	0.000000
1	0	0	0	0	5.18401	4.3	47	0.243222
2	0	0	0	0	3.57136	2.0	47	0.000000
3	0	0	0	0	3.45363	1.2	47	0.000000
4	0	0	0	0	3.54101	1.1	47	0.776676
...	...	...	...	...	...	...	...	...
9330	1070	92	1229056	112658	5.32973	4.2	45	0.000000
9331	878	80	1044812	99882	5.00407	4.4	44	0.832056
9332	1047	99	1207739	120956	4.80688	4.5	45	0.819128
9333	795	75	999803	92927	4.58043	1.2	44	0.000000
9334	543	45	725440	64427	4.66026	1.2	44	1.364530

Fig. 3. Dataset

Figure 4 depicts the correlation matrix of all features within the dataset. The analysis reveals that threads exhibit negligible influence and demonstrate minimal impact on other features. Conversely, a robust correlation is observed between the count of messages read by the CPU and the count of messages sent. Furthermore, the number of sent messages exhibits a significant correlation with the volume of sent bytes. Notably, a pronounced correlation emerges between the volume of sent bytes and CPU power, underscoring their interdependence.

Probability Density Function (PDF) was applied on the CPU Power. The PDF describes the relative likelihood of observing different values of a continuous random variable within a given range. In other words, the PDF provides a way to quantify the probability distribution of continuous random variables. Figure 5 shows that the CPU Power data has positively skewed distribution. There's a discrepancy between the median and the mean of the CPU power. The median, which represents the middle value of a dataset when it is arranged in ascending order, is 4.87. On the other hand, the mean, also known as the average, is 5.82.

Figure 6 shows the changes in the CPU power consumption. The range of CPU power consumption values spans from 3.02

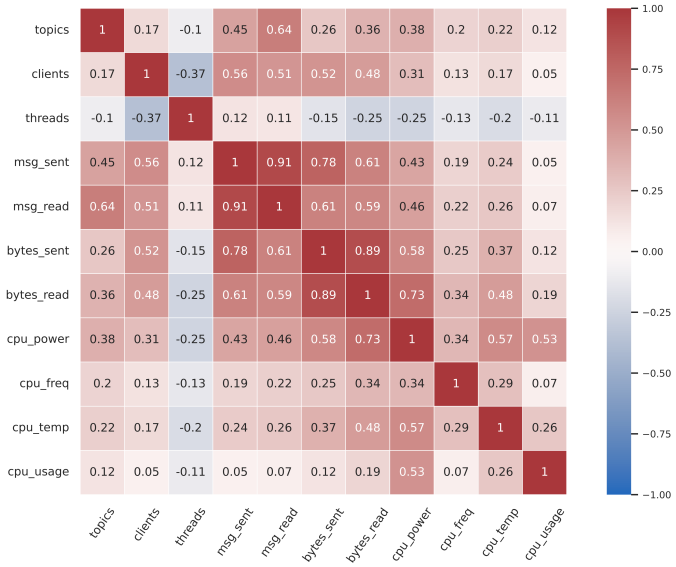

Fig. 4. Correlation Matrix of the Features in the Dataset.

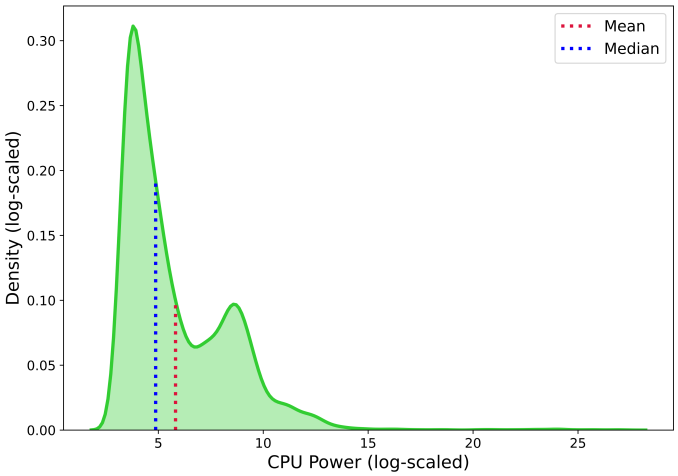

Fig. 5. Probability Density Function of the CPU Power.

to 27.01, indicating the full extent of variability in the dataset. The third quartile, representing the 75th percentile of the data, is 7.54. This indicates that 75% of the data points have a CPU power consumption below 7.54.

B. Methods

We applied different machine learning regression algorithms on the data and compared their results.

1) Linear Models:

a) Linear Regression: Linear regression [14] is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the independent variables (predictors) and the dependent variable (outcome). The main goal of linear regression is to find the best-fitting straight line that describes the relationship between the variables.

b) Polynomial Regression: Polynomial regression [15] is a type of regression analysis used to model the relationship

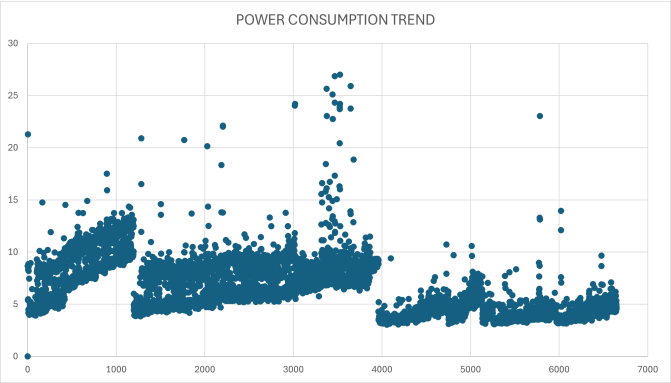

Fig. 6. Trend in CPU Power Consumption

between a dependent variable and one or more independent variables by fitting a polynomial function to the data. Unlike linear regression, which assumes a linear relationship between the variables, polynomial regression allows for more complex, nonlinear relationships to be captured.

c) Ridge Regression: Ridge regression [16] is a regularization technique used in linear regression to mitigate multicollinearity (high correlation among predictors) and reduce the model's sensitivity to the noise present in the data. It is particularly useful when dealing with datasets where the number of predictors (features) is large compared to the number of observations. In traditional linear regression, the goal is to minimize the sum of squared residuals between the observed and predicted values. However, when the dataset contains highly correlated predictors, the estimated coefficients can become large and unstable, leading to overfitting. Ridge regression addresses this issue by adding a penalty term to the traditional least squares objective function.

d) Lasso Regression: Lasso regression [17], short for Least Absolute Shrinkage and Selection Operator, is a regularization technique used in linear regression to address multicollinearity and perform feature selection by imposing a penalty on the absolute value of the coefficients. Similar to ridge regression, lasso regression adds a penalty term to the ordinary least squares (OLS) objective function. However, instead of penalizing the squared sum of coefficients as in ridge regression, lasso penalizes the absolute sum of coefficients.

2) Support Vector Machines:

a) Support Vector Regression: Support Vector Regression (SVR) [18] is a type of regression algorithm that utilizes the principles of support vector machines (SVM) to perform regression tasks. While traditional regression methods aim to minimize the error between the predicted and actual values, SVR focuses on fitting a hyperplane that best captures the relationship between the input variables and the target variable. The main idea behind SVR is to find a hyperplane in a high-dimensional feature space that has the maximum margin, while still minimizing the error between the predicted and actual values. This hyperplane is determined by support vectors, which are the data points closest to the hyperplane and influence its position and orientation. SVR operates by mapping the input variables into a high-dimensional feature space using a kernel function, which allows SVR to handle non-linear relationships between the variables. Once the data is transformed, SVR finds the hyperplane that best separates the data points, while also minimizing the error within a certain margin, known as the epsilon-insensitive tube.

b) Nu Support Vector Regression: Nu Support Vector Regression (Nu-SVR) is a variant of Support Vector Regression (SVR). NuSVR introduces a parameter ν (nu) that offers more control over the number of support vectors and the flexibility of the regression model.

3) Decision Trees: A Decision Tree for regression [19] is a predictive modeling algorithm that partitions the feature space into a set of simple decision rules, represented as a tree-like structure. Unlike classification trees that predict categorical outcomes, decision trees for regression predict continuous numerical values. The construction of a decision tree involves recursively splitting the data into subsets based on the feature that best separates the data according to some criterion. The process aims to minimize the variance of the target variable within each subset. At each split, the algorithm evaluates all possible splits on each feature and selects the one that results in the largest reduction in variance, often measured by metrics like mean squared error (MSE). Once a split is made, the process continues recursively on each subset until a stopping criterion is met. This could be a maximum tree depth, a minimum number of samples required to split a node, or when further splitting does not lead to a significant reduction in variance.

4) Ensemble Methods: Ensemble methods leverage a fusion of diverse algorithms, amalgamating their individual predictions. The primary objective is to enhance prediction accuracy, distinguishing ensemble methods from conventional model selection techniques. Nonetheless, the integration of various analytical approaches and the comparative assessment of their outcomes serve as a means to glean deeper insights into algorithmic behavior, particularly in scenarios where labels are absent.

a) Random Forest: Random Forest for regression [20] is a variant of the Random Forest algorithm designed specifically for predicting continuous numerical values. It operates by constructing an ensemble of decision trees, where each tree predicts a numerical value for a given input, and the final prediction is obtained by averaging the predictions of all trees in the forest.

b) Gradient Boosting: Gradient Boosting for regression [21] is an ensemble learning technique used to build predictive models for continuous numerical values. It works by sequentially adding weak learners (typically decision trees) to an ensemble, with each new learner correcting the errors made by the previous ones.

c) Bagging Regression: Bagging regression [22], short for Bootstrap Aggregating regression, is an ensemble learning technique used for building predictive models for regression tasks. It works by creating multiple bootstrap samples (random

samples with replacement) from the original dataset and training a separate base learner (regression model) on each sample. The final prediction is obtained by averaging the predictions of all base learners.

d) Voting Regression: A voting regressor [23] is an ensemble meta-estimator used in regression tasks, which fits multiple base regressors on the entire dataset and then averages their individual predictions to generate a final prediction. It operates similarly to a voting classifier in ensemble learning but is tailored for regression problems. By combining predictions from various regression models through simple or weighted averaging, voting regression enhances prediction accuracy and robustness, making it a powerful technique in machine learning for regression tasks.

e) AdaBoost: AdaBoost Regression [24] is a variant of the AdaBoost ensemble learning algorithm tailored for regression tasks. It combines the predictions of multiple base regression models, typically decision trees with limited depth (weak learners), to make a final prediction.

f) K-Nearest Neighbors: K-Nearest Neighbors (KNN) Regression [25] is a non-parametric supervised learning algorithm used for regression tasks. Unlike parametric models that learn a fixed number of parameters from the data, KNN Regression makes predictions based on the average of the target variable values of the K-nearest neighbors to a given query point.

g) Radius Neighbors Regression: Radius Neighbors Regression is a non-parametric regression algorithm that falls under the category of instance-based learning. Instead of selecting a fixed number of nearest neighbors (as in KNN), Radius Neighbors Regression selects all data points within a specified radius (known as the "radius parameter") around the query point.

C. Implementation and Evaluation

We applied dimensionality reduction using Principal Component Analysis (PCA) on the features' values. After that, we applied the above mentioned regression algorithms on the data. Table I shows a comparison between the results of applying different regression algorithms. The evaluation of regression models involves assessing how well the model's predictions match the actual values of the target variable. Several metrics can be used to measure the performance of regression models, depending on the specific goals and characteristics of the dataset. We decided to evaluate the results of the regression by three different matrices to have better overview of the performance.

Adjusted R-squared (R^2): R-squared measures the proportion of the variance in the target variable that is explained by the regression model. It ranges from 0 to 1, with higher values indicating better model fit. Adjusted R-squared is a modified version of R-squared that penalizes the addition of unnecessary predictors to the model. It provides a more reliable measure of model fit when comparing models with different numbers of predictors.

Root Mean Squared Error (RMSE): RMSE is the square root of the MSE and represents the average deviation of predicted values from the actual values. It is in the same units as the target variable and provides a more interpretable measure of prediction error.

Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted values and the actual values. It is less sensitive to outliers compared to MSE and provides a measure of the average magnitude of prediction errors.

Method	**Adjusted R-Squared (R^2)**	**Root Mean Squared Error**	**Mean Absolute Error**
Linear	0.69	2.05	1.04
Polynomial	0.70	1.96	1.01
Ridge	0.69	2.05	1.04
Lasso	0.20	5.23	1.78
SVR	0.01	6.46	1.58
NuSVR	0.25	4.92	1.63
Decision Tree	0.70	2.00	0.86
Random Forest	0.78	1.47	0.75
Gradient Boosting	0.74	1.71	0.86
Bagging	0.71	1.92	0.89
Voting	0.77	1.49	0.74
AdaBoost	0.67	2.17	1.07
K-Nearest Neighbors	0.78	1.43	0.63
Radious Neighbors	0.78	1.47	0.63

TABLE I
COMPARISON OF THE RESULTS OF ALL THE APPLIED METHODS WITH DIEMNSIONALITY REDUCTION.

We applied the same regression algorithms on the data without dimensionality reduction. Table I demonstrates the results. As the table shows, the results of regression without using dimensionality reduction got better.

Method	**Adjusted R-Squared (R^2)**	**Root Mean Squared Error**	**Mean Absolute Error**
Linear	0.75	1.63	0.92
Polynomial	0.79	1.35	0.81
Ridge	0.75	1.63	0.92
Lasso	-0.00	6.57	2.02
SVR	0.70	1.97	0.92
NuSVR	0.71	1.91	0.94
Decision Tree	0.79	1.40	0.66
Random Forest	0.85	0.96	0.55
Gradient Boosting	0.84	1.06	0.61
Bagging	0.81	1.22	0.68
Voting	0.86	0.93	0.53
AdaBoost	0.73	1.75	0.95
K-Nearest Neighbors	0.85	0.99	0.48
Radious Neighbors	0.82	1.19	0.55

TABLE II
COMPARISON OF METHODS WITHOUT DIMENSIONAL REDUCTION

The Voting algorithm emerges as the top performer among all applied algorithms, boasting an impressive Adjusted R-Squared of 0.86, a Root Mean Squared Error (RMSE) of 0.93, and a Mean Absolute Error (MAE) of 0.53. The residuals have been calculated by subtracting the ground truth CPU power values and the predicted values by the Voting algorithm. Additionally, the description of the Residuals Frequency plot (Figure 7) indicates that the residuals (the differences between predicted and actual values) follow an approximately Gaussian distribution, with most of them clustered around zero. This is a desirable characteristic as it indicates that the model's predictions are generally close to the actual values, with relatively few large errors. Overall, these results suggest that the Voting algorithm is performing well in accurately predicting CPU power values.

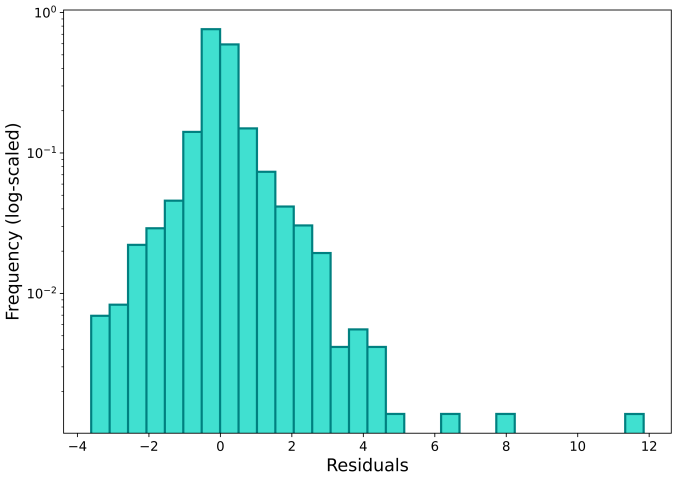

Fig. 7. The residuals of the output of the Voting regression.

Figure 8 depicts the Q-Q plot of the Residuals. In a Q-Q plot, each point represents a comparison between the theoretical quantile and the corresponding sample quantile. If the residuals follow a normal distribution, these points should fall along a straight line. Deviations from this line indicate departures from normality in the distribution of residuals.

The prediction outcomes generated by the voting algorithm are illustrated in Figure 9. This figure provides a visual representation of the comparison between the prediction outputs produced by the voting algorithm and the actual ground truth values within the test dataset.

D. Challenges

Predicting power consumption using machine learning involves several technical challenges, spanning Data Collection and Quality, Feature Engineering, Handling Seasonality and Trends, Model Selection and Training, Real-time Prediction and Scalability, Evaluation and Validation, Deployment and Maintenance, External Factors and Uncertainty, and Regulatory and Ethical Considerations. Addressing these challenges requires a combination of advanced data processing techniques, robust machine learning methodologies, and continuous model evaluation and improvement strategies.

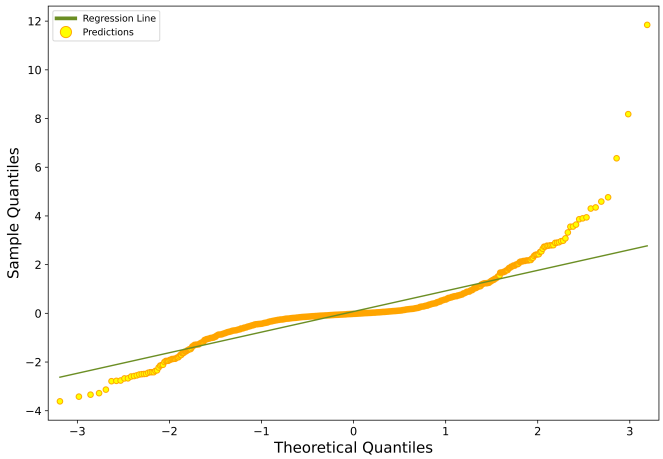

Fig. 8. Q-Q plot of the Residuals.

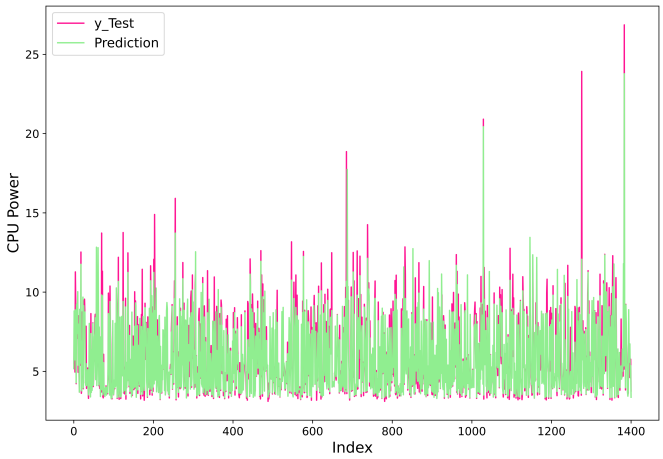

Fig. 9. Prediction vs. Ground Truth

V. Conclusion and Future Work

Based on the provided scientific paper, it presents an in-depth analysis and predictive model for the power consumption of an IoT broker operating with a Publish/Subscribe (Pub/Sub) messaging pattern. The paper integrates various parameters such as message traffic, number of clients, threads, topics, etc., into a cohesive framework to accurately estimate power consumption. Through empirical validation and simulation, the efficacy of the approach is demonstrated, achieving up to 85% prediction accuracy under varying workload scenarios. In conclusion, the research presented in this paper addresses the critical need for understanding and predicting the power consumption of IoT brokers, particularly in the context of Pub/Sub messaging patterns. By leveraging machine learning techniques and incorporating key parameters, the developed model offers valuable insights for resource allocation, anomaly detection, and optimization strategies within the Open6GHub initiative. The empirical validation and simulation results underscore the effectiveness of the approach in accurately predicting power consumption, thus contributing to the ad-

vancement of energy-efficient communication systems.

While the current research lays a solid foundation for predicting power consumption in IoT brokers, several avenues for future exploration and enhancement remain open. Some potential areas for future work include:

Enhanced Model Complexity: Investigating more complex machine learning models, such as deep learning architectures, to capture intricate relationships between input parameters and power consumption. These models may offer improved accuracy and robustness, particularly in scenarios with high-dimensional data or non-linear relationships.

Dynamic Model Adaptation: Developing adaptive models that can dynamically adjust their parameters based on evolving workload patterns and environmental conditions. This would enable real-time optimization of power consumption and resource allocation, leading to more responsive and energy-efficient IoT broker systems.

Integration with Dynamic Power Management: Exploring synergies between the predictive power consumption model and dynamic power management techniques, such as DVFS or task scheduling algorithms. By combining predictive insights with proactive power management strategies, IoT broker systems can achieve even greater energy efficiency without compromising performance.

Validation in Real-World Deployments: Conducting extensive field trials and validation studies in real-world IoT deployments to assess the practical effectiveness and scalability of the proposed predictive model. This would involve testing the model across diverse scenarios, hardware configurations, and environmental conditions to validate its generalizability and applicability in real-world settings.

Security and Privacy Considerations: Addressing security and privacy implications associated with collecting and analyzing sensitive data related to IoT communication and power consumption. Developing robust security mechanisms and privacy-preserving techniques to safeguard data integrity and user privacy while still enabling effective power consumption prediction and optimization.

By pursuing these avenues for future work, researchers can further advance the state-of-the-art in energy-efficient IoT communication systems and contribute to the broader goal of sustainable and resilient network infrastructure.

ACKNOWLEDGMENT

The authors acknowledge the financial support by the German *Federal Ministry for Education and Research (BMBF)* within the project Open6GHub {16KISK003K}.

REFERENCES

1 Pouhela, F., Krummacker, D., and Schotten, H. D., "Towards 6G Networks," in *A Context Management Architecture for Decoupled Acquisition and Distribution of Information in Next-Generation Mobile Networks*, ser. ITG, vol. 157, VDE. IEEE, 5 2023.

2 OASIS. Mqtt version 5.0, oasis standard. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

3 Pouhela, F., Sanon, S. P., Krummacker, D., and Schotten, H. D., "Everything Interconnected via Cyberspace," in *MMQP: A Lightweight, Secure and Scalable IoT Communication Protocol*. IEEE, 8 2024.

4 Chen, Y., Gao, J., Guo, Y., and Chen, H., "A machine learning-based approach for power consumption optimization in cloud data centers," *IEEE Access*, vol. 7, pp. 168 782–168 792, 2019.

5 Liu, Y., Chen, Y., Gao, J., and Chen, H., "Reinforcement learning-based approach for power consumption optimization in cloud data centers," in *2018 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. IEEE, 2018, pp. 636–641.

6 Pouhela, F., Krummacker, D., and Schotten, H. D., "Entity component system architecture for scalable, modular, and power-efficient iot-brokers," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1–6.

7 Chang, C.-H., Chen, H.-H. S., and Hsu, Y.-C., "Dynamic voltage and frequency scaling techniques: A survey," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 8, pp. 1792–1806, 2007.

8 Kim, Y.-S. and Lee, K., "Dynamic voltage and frequency scaling techniques for low power," *ACM Computing Surveys (CSUR)*, vol. 41, no. 4, pp. 1–36, 2009.

9 Wang, H., Zhang, L., He, X., and Chen, H.-H. S., "Dynamic voltage and frequency scaling techniques: A survey," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, pp. 1–36, 2011.

10 Bapatla, S. and Raghunathan, A., "Power-aware scheduling of multi-threaded applications on multicore processors," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 11, no. 4, pp. 1–30, 2014.

11 Mao, K. and Chen, Y., "Power-aware task scheduling for multi-core processors: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 1, pp. 1–38, 2016.

12 Jeon, M.-K. and Kim, K., "Power-aware task scheduling for multi-core processors: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 1, pp. 1–37, 2017.

13 "Asio c++ library." [Online]. Available: https://think-async.com/Asio/

14 Maulud, D. and Abdulazeez, A. M., "A review on linear regression comprehensive in machine learning," *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 140–147, 2020.

15 Heiberger, R. M., Neuwirth, E., Heiberger, R. M., and Neuwirth, E., "Polynomial regression," *R Through Excel: A Spreadsheet Interface for Statistics, Data Analysis, and Graphics*, pp. 269–284, 2009.

16 Hoerl, R. W., "Ridge regression: a historical context," *Technometrics*, vol. 62, no. 4, pp. 420–425, 2020.

17 Ranstam, J. and Cook, J. A., "Lasso regression," *Journal of British Surgery*, vol. 105, no. 10, pp. 1348–1348, 2018.

18 Zhang, F. and O'Donnell, L. J., "Support vector regression," in *Machine learning*. Elsevier, 2020, pp. 123–140.

19 Tso, G. K. and Yau, K. K., "Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks," *Energy*, vol. 32, no. 9, pp. 1761–1768, 2007.

20 Segal, M. R., "Machine learning benchmarks and random forest regression," *Center for Bioinformatics and Molecular Biostatistics*, 2004.

21 Nie, P., Roccotelli, M., Fanti, M. P., Ming, Z., and Li, Z., "Prediction of home energy consumption based on gradient boosting regression tree," *Energy Reports*, vol. 7, pp. 1246–1255, 2021.

22 Altman, N. and Krzywinski, M., "Ensemble methods: bagging and random forests," *Nature Methods*, vol. 14, no. 10, pp. 933–935, 2017.

23 Lindner, C. and Cootes, T. F., "Fully automatic cephalometric evaluation using random forest regression-voting," in *IEEE International Symposium on Biomedical Imaging*, vol. 13. Citeseer, 2015.

24 Shrestha, D. L. and Solomatine, D. P., "Experiments with adaboost. rt, an improved boosting scheme for regression," *Neural computation*, vol. 18, no. 7, pp. 1678–1710, 2006.

25 Kramer, O., "Unsupervised k-nearest neighbor regression," *arXiv preprint arXiv:1107.3600*, 2011.