actuators

Article

A Feature Selection-Based Predictive-Learning Framework for Optimal Actuator Control in Smart Homes

Sehrish Malik [1], Wafa Shafqat [1], Kyu-Tae Lee [1,*] and Do-Hyeun Kim [2,*]

[1] Division of Information and Communication Engineering, Kongju National University, Cheonan 331717, Korea; serrym29@gmail.com (S.M.); wafashafqat92@gmail.com (W.S.)
[2] Computer Engineering Department, Jeju National University, Jeju-si 63243, Korea
* Correspondence: ktlee@kongju.ac.kr (K.T.L.); kimdh@jejunu.ac.kr (D.K.)

Abstract: In today's world, smart buildings are considered an overarching system that automates a building's complex operations and increases security while reducing environmental impact. One of the primary goals of building management systems is to promote sustainable and efficient use of energy, requiring coherent task management and execution of control commands for actuators. This paper proposes a predictive-learning framework based on contextual feature selection and optimal actuator control mechanism for minimizing energy consumption in smart buildings. We aim to assess multiple parameters and select the most relevant contextual features that would optimize energy consumption. We have implemented an artificial neural network-based particle swarm optimization (ANN-PSO) algorithm for predictive learning to train the framework on feature importance. Based on the relevance of attributes, our model was also capable of re-adding features. The extracted features are then applied as input parameters for the training of long short-term memory (LSTM) and optimal control module. We have proposed an objective function using a velocity boost-particle swarm optimization (VB-PSO) algorithm that reduces energy cost for optimal control. We then generated and defined the control tasks based on the fuzzy rule set and optimal values obtained from VB-PSO. We compared our model's performance with and without feature selection using the root mean square error (RMSE) metric in the evaluation section. This paper also presents how optimal control can reduce energy cost and improve performance resulting from lesser learning cycles and decreased error rates.

Keywords: actuator control; energy prediction; context-aware; feature selection; sensing; actuator control; smart buildings

check for updates

Citation: Malik, S.; Shafqat, W.; Lee, K.T.; Kim, D. A Feature Selection-Based Predictive-Learning Framework for Optimal Actuator Control in Smart Homes. *Actuators* **2021**, *10*, 84. https://doi.org/10.3390/act10040084

Academic Editors: Ioan Ursu and Hicham Chaoui

Received: 25 February 2021
Accepted: 14 April 2021
Published: 20 April 2021

1. Introduction

In recent years, household consumption of energy has drastically increased due to the rapid growth rate observed in the world's population. With the increase in population, houses have also expanded, and the use of appliances has become more common. Household appliances such as air conditioners, heaters, refrigerators, washing machines, stoves, etc., all operate on energy. Early prediction of a household's energy usage can help in better managing of the energy needs and planning to save energy where possible. Hence, predicting electricity demand is crucial, as it plays a pivotal role in utility power planning [1]. Effective and accurate energy consumption models cannot be overemphasized and are one of the major challenges [2,3].

A context-aware energy prediction mechanism is one where roles of all the sensing values from the environment surroundings are carefully observed. For example, in a smart home's energy consumption prediction problem, the context can include the smart home appliances' power consumption, weather conditions, users' activities, time of the day, day of the week, workday or holiday, special events, etc. A prediction mechanism must consider the surroundings of the given environment and the fact that the surrounding relevance is also variable. Another major concern while implementing a prediction model is selecting the most appropriate features.

Feature selection is one of the most important steps in prediction problems. The process of feature selection can be defined as finding the smallest subset that shows the strongest effect on the prediction accuracy and minimizes the model's complexity. Finding the most related input features is essential. To select the right features, one must have detailed and in-depth knowledge of the area. Still, manual feature selection is a very tedious task, and even experts in a field can make mistakes. The accuracy of the prediction model greatly depends on the quality of data and the relevancy of features. For example, in case of energy prediction, if the given features in a dataset have no strong relation to the increase or decrease in the energy consumption, then there are high chances that the model performance will turn out to be poor. A prediction model, which is enabled to learn feature importance on its own, with the passage of time, can be of huge benefit to prediction problem applications.

In this work, we focused on the use of a long short-term memory (LSTM) algorithm in prediction models. LSTM is a type of recurrent neural network (RNN). It is a time-series forecasting algorithm [4]. LSTM networks are considered one of the most suitable prediction algorithms for time series data. In recent years, many researchers have focused on proposing prediction algorithms using the efficacy of LSTMs. Xiangyun et al. propose an hourly day-ahead solar irradiance prediction algorithm, using LSTMs, for minimizing energy costs [5]. An energy consumption prediction mechanism for smart homes based on a hybrid LSTM network is proposed by Ke et al. [6]. The hybrid approach increases the data dimensions and eventually results in increased prediction accuracy.

In this work, we present a predictive learning-based optimal actual control mechanism for smart homes. The proposed mechanism includes a prediction learning module that uses LSTM to make energy consumption predictions. The prediction module takes all available features at the first cycle of learning and works its way down to learn the most impactful features. The prediction model also learns to re-add features based on the relevance of history learned at a given prediction time. The predicted values are then passed onto the optimization module, where optimal parameters are set accordingly. Optimal parameters are then used to generate actuator control commands.

The rest of the paper is divided as follows: Section 2 presents the related works; Section 3 presents the proposed prediction mechanism. In Section 4, we provide the task modeling simulation of the proposed system. Results analysis is presented in Section 5; Section 7 concludes the paper with discussions.

2. Related Work

Energy consumption prediction is one of the significant prediction problems, and many recent researchers have proposed solutions for energy prediction based on deep learning, such as load forecast-based on pinball loss guided LSTM [7], energy use prediction for solar-assisted water heating system [8], and short-term residential load forecasting using LSTM recurrent neural network [9,10]. An LSTM-based periodicity energy usage of a cooling system is analyzed in [11]. According to this study, the computation cost can be reduced by using lower dimensional variables. A hybrid LSTM-based approach that integrated data pretreatment strategy, deep learning method, and advanced optimization method, is introduced in [12]. It is referred as a short-term energy load forecast system known as VMD-BEGA-LSTM (VLG), integrating a data pretreatment strategy, advanced optimization technique, and deep learning structure, is developed in this paper.

LSTMs have been widely used in contextual recommendation systems. A context-aware location recommendation system is proposed by Wafa et al. The authors propose a hierarchical LSTM model. The two-level hierarchy model predicts the location of interest at the first level and considers the contextual information related to predicted location at the second level [13].

Yirui et al. propose a context-aware attention LSTM network model for flood prediction. The authors focus on predicting sequential flow rates based on flood factors. Their proposal aims to remove the irrelevant flood factors data to avoid noise and emphasize on the informative factors only. Their proposed model learns the probability distributions between flow rate and hidden output of each LSTM during training and assigns the weights accordingly during testing [14]. The installment of IoT sensors in smart building enables action recognition and hence, it leads to real-time monitoring, control, and savings of energy. A context-aware load supply mechanism based on IoT and deep learning is presented. The context-aware network is built on classroom video and action recognition data is extracted to extract contexts [15]. Joshua et al. present a hot water energy demand prediction for saving energy using convolutional neural networks (CNNs). They make the use of contextual data such hour, day, week, etc., to extract contextual information [16].

Jeong et al. present a context-aware LSTM for event time series prediction. Their proposed model is based on hidden Markov model (HMM) and LSTM. HMM is used to abstract the past data for distant information of context [17]. Maria et al. make an effort to cover two properties of non-intrusive load monitoring, non-causality and adaptivity to contextual factors, via application of bidirectional LSTM model. The authors develop a self-training-based adaptive mechanism to address scaling issues with the increase in smart home appliances [18]. Another scalable and non-intrusive load monitoring approach is presented by Kunjin et al. A convolutional neural network (CNN)-based model is proposed for building a multi-branch architecture, with an aim to improve prediction accuracies [19]. Tae et al. present a CNN-LSTM-based hybrid mechanism for household power consumption prediction [20]. Yan et al. present a CNN-LSTM-based integrated energy prediction approach for Chinese energy structure. In order to verify the results, the authors compared their proposal results with six methods such as ARIMA, KGM, SVM, PBNN, LSTM, and CNN [21]. Many recent works have proposed hybrid of popular prediction algorithms to present a more robust solution [22–26].

Many recent studies have focused on the context-aware load prediction solutions. The existing solution of context-aware predictions can be widely categorized into two types: context feature extraction in pre-processing and context features weights assignment during learning. Based on our related works' study, we observed that a major difference in the prediction results might occur; if past predictions' context is recorded, and feature learning is performed before weighing the features during training process. A summary of context-aware prediction solutions is presented in Table 1.

Table 1. Summary of context-aware prediction solutions.

Ref.	Feature Extraction in Pre-Processing	Feature Extraction During Prediction Learning	Feature Learning Through Prediction Learning Phases
[13]	√	-	-
[14]	√	√	-
[15]	√	√	-
[16]	√	√	-
[17]	√	√	-
[18]	√	√	-
[19]	√	√	-
[20]	√	-	-
[21]	√	-	-
Proposed Approach	√	√	√

Feature learning and selection based on feature importance is also very crucial when it comes to learning contexts and updating model accordingly. In [27], an integrated feature learning technique is proposed for predicting travel time by using deep learning. To increase the learnability of features, different feature enriching algorithms are applied. In [28], the focus is on selecting effective features through a strategy that minimizes the redundancy and maximizes the relevancy of features. The study analyzes the influence of selecting effective features on load consumption of building. In another study [29], deep learning and Bat algorithms are used for optimizing energy consumption and for user preference-based feature selection. A hybrid approach [30] that is integrated with feature selection method aims to automatically detect the features based on higher relevance for multi-step prediction. In [31], two main tasks are performed: first, a method is proposed to transform the time-dependent data for machine learning algorithms and secondly, different kind of feature selection tasks are applied for regression tasks.

A similar sort of model is presented in [32], that is developed by using layered structures and meta-features. In [33], a neuro-fuzzy inference system is proposed that is being boosted with an optimizer. It also introduced a new non-working time adaptation layer. In [34], an LSTM-based model with various configurations is built to forecast energy consumption. It uses wrapper and embedded feature selection techniques along with genetic algorithm (GA) to find optimal configuration of LSTM for prediction learning.

Prediction learning models are updated periodically, due to updates in history data. Hence, it is wise to record additional results from previous test results and use the data to benefit the system in next training cycles. In this work, we propose a feature learning solution that aims to improve the prediction accuracies by filtering the contextual data based on history learnings.

3. Proposed Prediction Mechanism

This section presents a feature learning-based adaptive LSTM approach for energy predictions in smart environments. Our proposed prediction model takes all available features at the first cycle and works its way down to learn the most impactful features. The proposed model also learns to re-add features based on relevance with a change in attributes involved and time. Hence, we can call the proposed model to be adaptive in nature.

In Figure 1, we present the proposed prediction model, where an LSTM algorithm is used for making predictions. The proposed model has a feature learning module integrated with LSTM predictions in a cyclic manner. After every cycle of LSTM prediction, the outputs are passed onto the feature learning module, where features are tuned, and then updated features are passed onto the next cycle of predictions.

3.1. LSTM Architecture

LSTM is a type of RNN (recurrent neural network) where prediction is performed by backpropagation. The output of a neural network layer is backpropagated at time "t" to the input of the same network layer at time 't + 1'.

LSTM contains memory blocks that are connected through layers. A block contains three gates that enable the block to maintain its states and output. The three gates included in an LSTM block are a forget gate, an input gate, and an output gate. The forget gate conditionally decides which information can be ignored and which information is important to be remembered. The input gate decides what values from input should be used to update the state. The output gate decides what should be the output based on the block's current input and state memory.

In equations flow below, we present the operations performed at forget gate, input gate, and output gate. The terms used are described in Table 2.

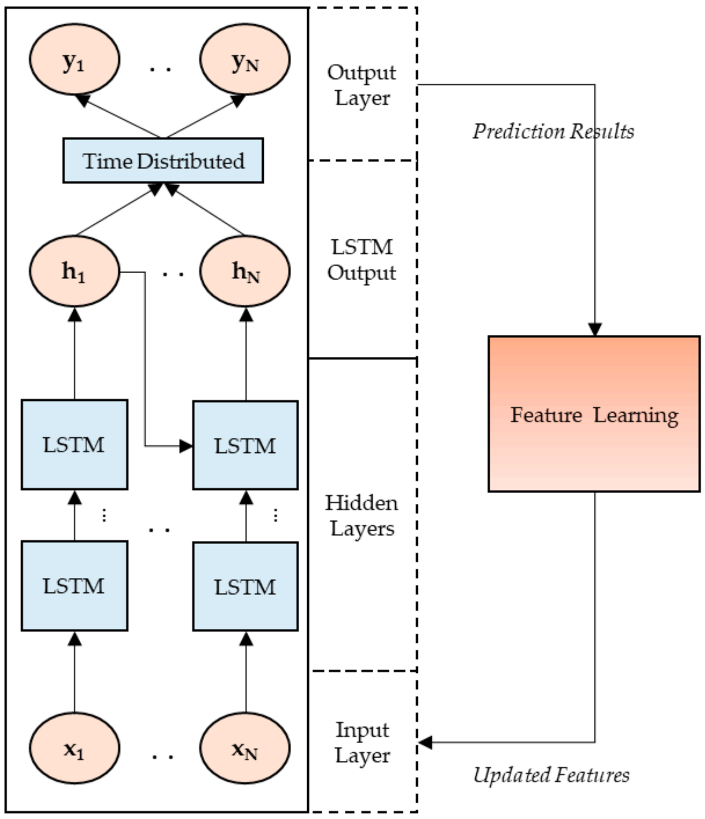

Figure 1. Proposed prediction model.

Table 2. Description of terms used at long short-term memory (LSTM) gate operations.

Terms	Description
σ	sigmoid function
$\tan h$	tanh function
t	Time step
f_t	Forget gate at t
x_t	Input at t
h_{t-1}	Previous hidden state
W_f	Weight matrix between forget gate and input gate
b_f	Connection bias at forget gate
i_t	Input gate at t
W_i	Weight matrix of sigmoid operator between input gate and output gate
b_i	Bias vector at input gate
C_t	Value generated by tanh operator
W_c	Weight matrix of tanh operator between cell state information and network output
b_c	Bias vector at t for weight matrix W_c
C_t	Cell state information
C_{t-1}	Previous time step cell state information
o_t	Output gate at t
W_o	Weight matrix of output gate
b_o	Bias vector for W_o
h_t	LSTM output

Equation (1) shows the operation of forget gate which decides what elements of the previous cell state (C_{t-1}) are to be forgotten.

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \quad (1)$$

Next, Equation (2) shows that which values is to be updated at input gate.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2}$$

Then a potential vector of cell state is computed by the current input (x_t) and the last hidden state h_{t-1}.

$$\mathrm{C}_t = \tan\mathrm{h}(W_c[h_{t-1}, x_t] + b_c) \tag{3}$$

After that, we can update the old cell state C_{t-1} into the new cell state C_t by element-wise multiplication as shown in Equation (4) below.

$$C_t = f_t * C_{t-1} + i_t * \mathrm{C}_t \tag{4}$$

The output gate decides which elements to output by a sigmoid layer (Equation (5)).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{5}$$

The new hidden state h_t of LSTM is then calculated by combining Equations (4) and (5).

$$h_t = o_t * \tan\mathrm{h}(C_t) \tag{6}$$

In Figure 2 below, we show the block structure of LSTM including forget gate, input gate, and output gate.

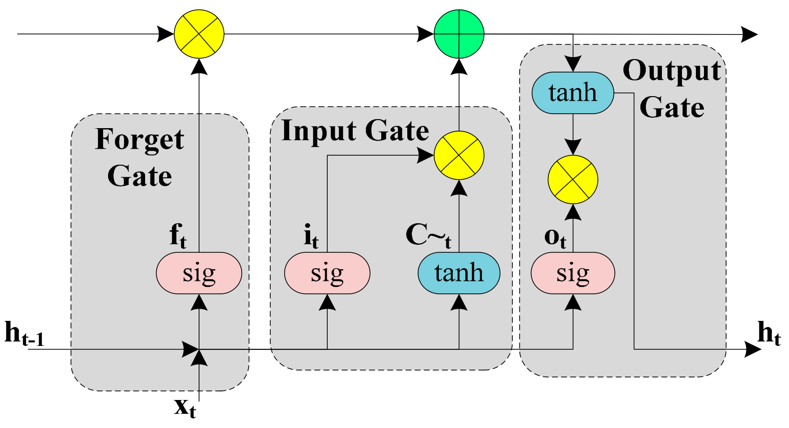

Figure 2. LSTM block structure.

3.2. Feature Learning

In this subsection, we present the feature learning method. The training and testing process using the LSTM algorithm is performed in cycles.

All features' relevance data is added to feature relevance history data. This history data is built by reiterating training and testing cycles, with feature deductions, shuffling, and reset. The aim of maintaining history log for feature relevance score is to make the system capable of learning from features relevance context based on time of predictions. The feature history data is updated with each cycle by adding prediction results attributes such as time of the day, day of the week, workday/holiday, and all features relevance score.

We have used an artificial neural network (ANN) algorithm for learning feature relevance, and we have used PSO algorithm to optimize the weights. PSO takes weights from ANN at each iteration, and PSO populations work towards finding optimal weights. The tuned weights are passed back to ANN. The optimization of ANN weights with PSO aims to reduce learning time and improve learning rates (Figure 3).

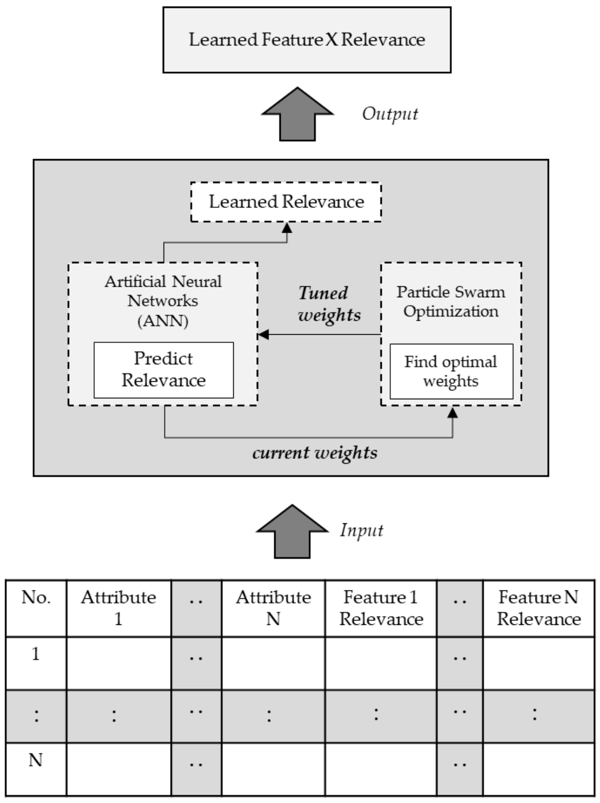

Figure 3. Overall feature learning process based on ANN and genetic algorithm (GA).

3.3. Meta-Parameters

This section presents some meta-parameters used during various processes and methods, such as different thresholds. These meta-parameters are set after multiple experiments, and the most optimal values are used. Other meta-parameters include total processing capacity, layers in ANN, PSO's inertia velocity boost threshold, and PSO's regeneration threshold. These thresholds are also used to examine the particle's performance in PSO. Table 3 presents the optimal values of different meta-parameters after multiple experiments, as shown in [35]. In different applications, users may have to tinker with these values to fine-tune their results.

Table 3. Meta-parameters.

Meta-Parameters	Optimal Values
Total Processing Capacity	95% of CPU available
ANN layers	6
PSO inertia velocity boost threshold	11
PSO re-generation threshold	25

After the first cycle of LSTM training and testing, each feature's prediction scores, and relevance score are extracted. A relevance threshold is set, and a relevance score below the threshold is considered unacceptable. After the first cycle, the least relevant feature is removed, and the next cycle of prediction is performed with updated input data. From the second iterations onwards, the system first validates the prediction performance after removal of the last least relevant feature. If prediction performance is improved or no change is observed, then validation is set to true. Otherwise, if prediction performance deteriorates after feature removal, then validation is set to false. When validation is true, the relevance score for all existing features is extracted, and the feature with the least relevance is removed. When validation is false, the last removed feature is added back, and a positive weight is added to its relevance history data (Figure 4).

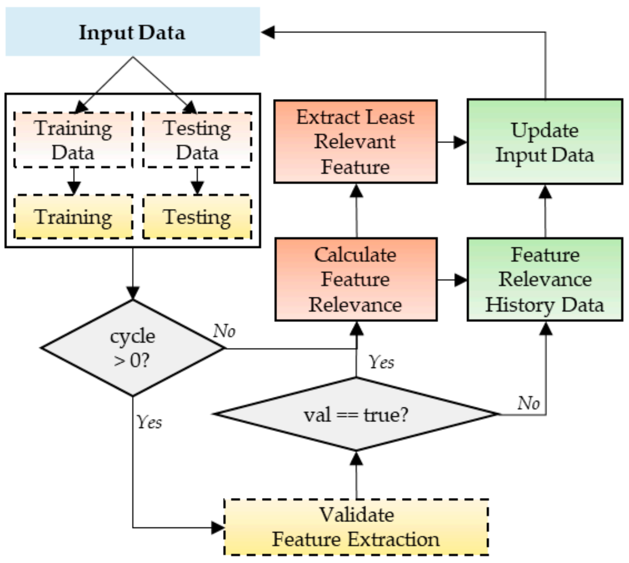
Input Data
Training Data
Testing Data
Training
Testing
Extract Least Relevant Feature
Update Input Data
Calculate Feature Relevance
Feature Relevance History Data
cycle > 0?
No
Yes
val == true?
Yes
No
Validate Feature Extraction

Figure 4. Feature relevance extraction and validation during cycle.

After the removal or addition of features based on validation results, the history data learning results are extracted. Features with high relevance score based on learnings from history data are matched to be present in the data, and features with relevance below than a set threshold are removed.

3.4. Predictive Learning-Based Optimal Control Mechanism

In this section, we present the predictive learning-based control mechanism for smart home actuators' control commands generation and scheduling the control commands. We upgrade the optimal control scheduling mechanisms from one of our recently published works [36] and transform it into a predictive learning-based optimal control mechanism to improve the architecture design and performance. In [36], we defined the optimization objective function based on user-defined parameters. In this proposal, the optimization function achieves the optimal values based on the demand forecasting from the prediction module instead of user-defined parameters (Figure 5). This enables the optimization objective function to be adaptive based on history learnings and makes the function less dependent on the user inputs. It will learn the user preferences based on prediction learnings.

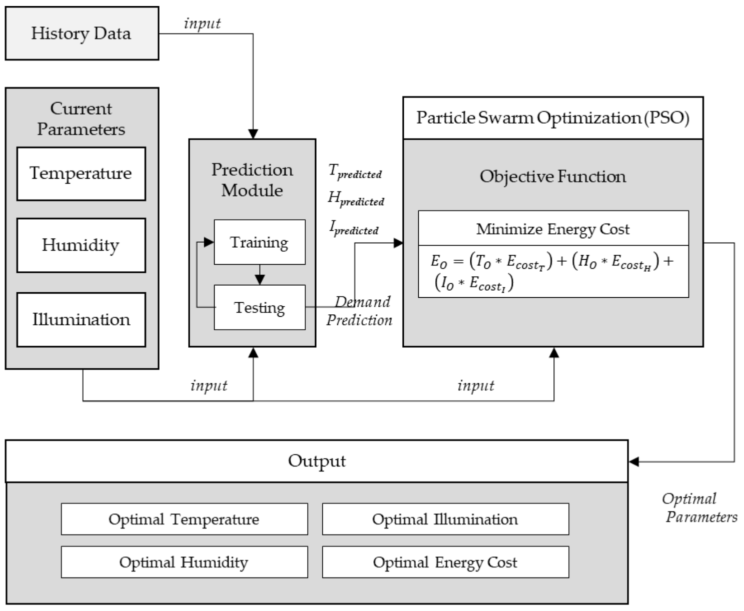

Figure 5. Prediction learning-based optimization mechanism.

First, the prediction module outputs the temperature, humidity, and illumination demand for the current timestamp. The predicted values are passed as input to the optimization module along with current parameters. The notations description for the optimization objective function is given in Table 4.

Table 4. Objective function notations description.

Notations	Description
$T_{Predicted}$	Predicted temperature energy demand values
$H_{Predicted}$	Predicted humidity energy demand values
$I_{Predicted}$	Predicted illumination energy demand values
T_C	Temperature reading at the current hour
H_C	Humidity reading at the current hour
I	Illumination reading at the current hour
TDD	Temperature demand difference calculated between predicted and current
HDD	Humidity demand difference calculated between predicted and current
IDD	Illumination demand difference calculated between predicted and current
E_{cost_T}	Per unit energy consumption cost for temperature
E_{cost_H}	Per unit energy consumption cost for humidity
E_{cost_I}	Per unit energy consumption cost for illumination
ECS_O	Optimal energy consumption savings
T_O	Optimal temperature
H_O	Optimal Humidity
I_O	Optimal Illumination
E_O	Optimal energy consumption

The equations flow given below, defines the optimization objective function for the three input parameters of temperature, humidity and illumination. The predicted temperature demand ($T_{Predicted}$), predicted humidity demand ($H_{Predicted}$) and predicted illumination demand ($I_{Predicted}$) values are taken as input from prediction module. Demand difference values for temperature (TDD), humidity (HDD), and illumination (IDD) are calculated by calculating the difference between the predicted values and the current values (Equations (7)–(9)). Next, energy demand difference cost is calculated (Equation (10)). Energy savings are calculated by running the optimization algorithm solution and finding the optimal parameters with minimized energy cost based on demand and current parameters (Equations (11) and (12)). Once the optimal parameters are found, the optimal energy can be calculated as given in Equation (13).

$$\text{TDD} = |T_{Predicted} - T_C| \tag{7}$$

$$\text{HDD} = |H_{Predicted} - H_C| \tag{8}$$

$$\text{IDD} = |I_{Predicted} - I_C| \tag{9}$$

$$EDD_{COST} = (TDD_{COST} * E_{cost_T}) + (HDD_{COST} * E_{cost_T}) + (IDD_{COST} * E_{cost_I}) \tag{10}$$

$$ECS_O = \ [E_{CT} + E_{CH} + E_{CI}] - EDD_{COST} \tag{11}$$

$$\begin{aligned} ECS_O = [E_{CT} + E_{CH} + E_{CI}] - [(|T_{Predicted} - T_O| * E_{cost_T}) \\ +(|H_{Predicted} - H_O| * E\cos t_H) + (|I_{Predicted} - I_O| * E\cos t_I)] \end{aligned} \tag{12}$$

$$E_O = (T_O * E_{cost_T}) + (H_O * E_{cost_H}) + (I_O * E_{cost_I}) \tag{13}$$

Once the optimal values are found, the optimal values are then used to derive fuzzy rules set to trigger the control commands based on input parameters to the fuzzy module. Table 5 explains the actuator actions constraints based on which fuzzy control module generates the control rules for each actuator action. The control commands generated from the fuzzy control module are then used to model control tasks which are to be executed at the scheduler for control tasks' operation (Figure 6). We have used our previously implemented fuzzy control and scheduling module [36] and made changes to fit our current scenarios.

Table 5. Actuator actions constraints for defining fuzzy control rules.

Data Attribute	Description
Heating	$T_c < T_{min} \leq T_o \leq T_{max}$
Chilling	$T_{min} \leq T_o \leq T_{max} < T_c$
Humidification	$H_c < H_{min} \leq H_o \leq H_{max}$
Dehumidification	$H_{min} \leq H_o \leq H_{max} < H_c$
Light-Brightness	$I_c < I_{min} \leq I_o \leq I_{max}$
Light-Darkness	$I \leq I_o \leq I_{max} < I_c$

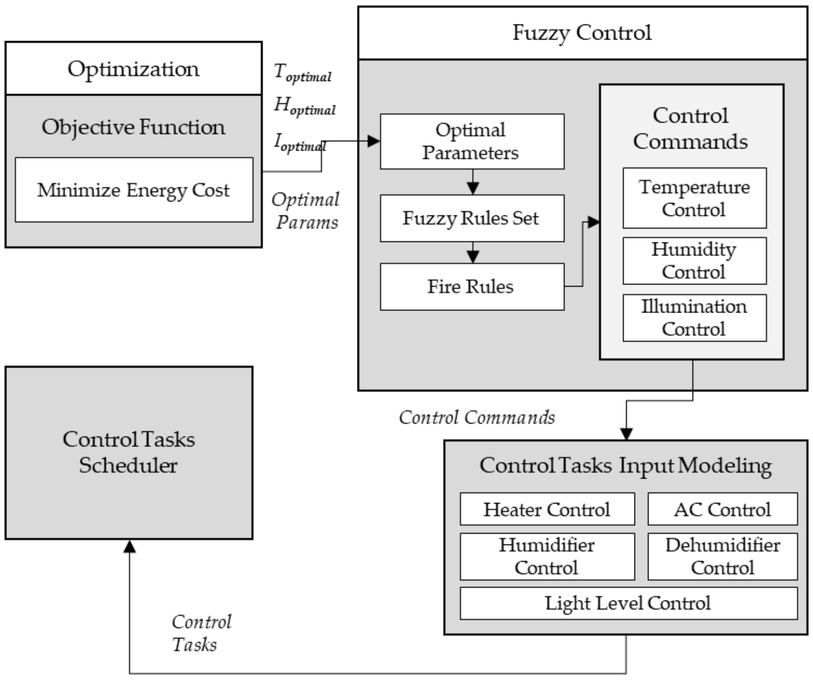

Figure 6. Prediction learning-based optimal control mechanism.

4. Dataset and Data Preprocessing

We have used dataset of energy consumption and related weather data [37]. The set contains timestamp, energy consumption, and weather conditions. The detailed data attributes are shown in Table 6.

Table 6. Description of terms used at LSTM gate operations.

No.	Data Attribute	Description
1	Timestamp	Hourly timestamp including year, month, day, hour for the readings
2	Energy	Energy consumption readings at the current hour
3	Conditions	Weather conditions such as rain or clear, fog, mist, partly cloudy, cloudy
4	Dew PointC	Dew point reading at the current hour
5	Gust speed	Gust speed reading at the current hour
6	Humidity	Humidity reading at the current hour
7	Temperature	Temperature speed reading at the current hour
8	Precipitation mm	Precipitation reading at the current hour
9	Sea Level Pressure	Sea level pressure reading at the current hour
10	Visibility Km	Visibility in kilometers at the current hour
11	Wind Speed Km/h	Speed of wind in kilometer per hour at the current hour
12	Wind Direction	Direction of wind at the current hour
13	WindDirDegrees	Degrees of wind direction at the current hour

Now, we present the data preprocessing steps as shown in Figure 7. History data was taken as input to model data. Data was analyzed using two data analysis techniques of visual assessment and programmatic assessment. Based on data analysis, the next step was to clean data. To tune the data format, we convert the timestamp data type to date time. Next, we resampled the weather data, as data should be resampled before merging energy data and weather data. In the last step, the data to be passed to the prediction module was normalized.

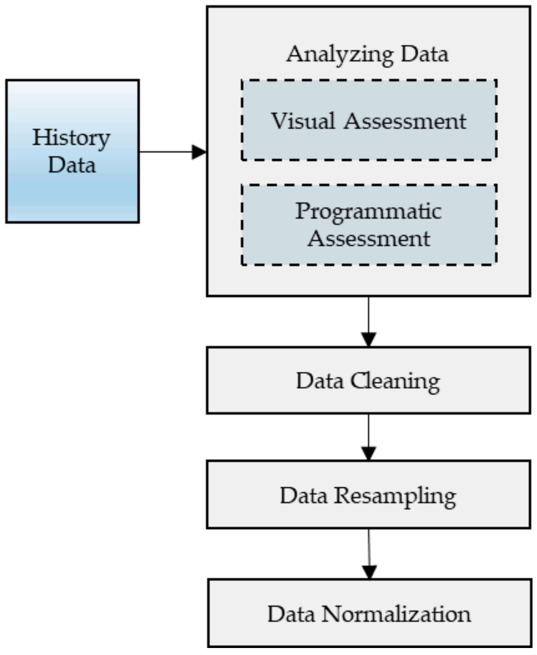

Figure 7. Input data preprocessing.

5. Implementation Setup

In this section, we present the implementation setup in detail. The core programming logic was implemented using Python 3.8.1. Python is a very popular general-purpose programming language; it is widely used for developing desktop and web-based applications. The implementation is mainly done on Jubyter Lab IDE (Integrated Development Environment) since, Jupyter lab provides ease of implementation, better results visualization, and high-level features to adapt to processing needs. The additional details of system configuration are presented in Table 7.

Table 7. Implementation environment.

Components	Specifications
Operating System	Windows 10 Professional Edition
Processor	Intel i5 9th Generation
Memory (RAM)	16 GB
Programming Language	Python 3.8.1
IDE	Jupyter (Conda 4.9.1)

6. Performance Analysis

In this section, first, we present the analysis of the results of predictions using multivariate LSTM algorithm with self-selected features. Then, we present the results analysis of predictions using multivariate LSTM algorithm with feature learning.

We use root mean square error (RMSE) measure as a performance metric to compare prediction performance. RMSE is the standard deviation of the residuals (prediction errors), where residuals are a measure of how far from the regression line data points are. In other words, it tells you how concentrated the data is around the line of best fit.

6.1. PredictionMechanism without Feature Learning

In this subsection, we perform predictions using self-selected contextual features. During the visual assessment and programmatic assessment, we observed the available features in the dataset. We studied the trends of each feature's data values over the period of year through different seasons. After carefully studying data trends, we decided to drop the features that seemed irrelevant to the data. Initially, we had 11 contextual features of temperature, humidity, dew point, conditions, gust speed, precipitation mm, sea level pressure, visibility km, wind speed km/h, wind direction, and wind direction degrees. After dropping irrelevant features, we were left with eight contextual features: temperature, humidity, dew point, precipitation mm, sea level pressure, visibility km, wind speed km/h, and wind direction degrees. We further ranked eight contextual features into most to least relevant based on our data trends study and analysis. The ranking in descending order was as following: temperature, humidity, dew point, precipitation mm, visibility km, wind speed, sea level pressure, and wind direction degrees (Table 8).

Table 8. Datasets based on contextual feature ranking.

		Data Columns
Dataset 1	1 Contextual Feature	Timestamp ∣ Energy ∣ Temperature
Dataset 2	2 Contextual Features	Timestamp ∣ Energy ∣ Temperature ∣Humidity
Dataset 3	3 Contextual Features	Timestamp ∣ Energy ∣ Temperature ∣Humidity ∣Dew PointC
Dataset 4	4 Contextual Features	Timestamp ∣ Energy ∣ Temperature ∣Humidity ∣Dew PointC∣ Precipitation mm
Dataset 5	5 Contextual Features	Timestamp ∣ Energy ∣ Temperature ∣Humidity ∣Dew PointC∣ Precipitation mm ∣ Visibility km
Dataset 6	6 Contextual Features	Timestamp ∣ Energy ∣ Temperature ∣Humidity ∣Dew PointC∣ Precipitation mm ∣ Visibility km ∣Wind Speed
Dataset 7	7 Contextual Features	Timestamp ∣ Energy ∣ Temperature ∣Humidity ∣Dew PointC∣ Precipitation mm ∣ Visibility km ∣Wind Speed ∣ Sea Level Pressure
Dataset 8	8 Contextual Features	Timestamp ∣ Energy ∣ Temperature ∣Humidity ∣Dew PointC∣ Precipitation mm ∣ Visibility km ∣Wind Speed ∣ Sea Level Pressure ∣ WindDirDegrees

Figure 8 shows the weekly prediction results of all eight datasets. We can observe from the results that performance deviation does not necessarily align with our feature selection order. Though in some cases, removal of an irrelevant contextual feature (based on self-ranking) has shown improvement in the prediction performance such as from Dataset 8 to Dataset 7, but right in the next removal, from Dataset 7 to 6, the performance was badly affected as well. Similarly, a mixed effect on the prediction performance was observed for training and testing using Dataset 6 until Dataset 1.

The average RMSE values for each dataset with a varying number of contextual features are given in Figure 9. Our ranking of contextual features still proved to be relevant as best prediction performance observed with Dataset 2 containing two top-ranked contextual features. Still, the best prediction performance in this scenario had considerable deviations of predicted values from the actual values. Unless we derived datasets of all the contextual features' combinations and trained and tested our prediction model with all datasets, we cannot be sure that the achieved results with the given Dataset 2 were the best results possible. Such regressive training with all the combinations of features still seems a doable task for datasets with fewer features; but as the number of features increases, it becomes very costly to train and test for all the possible combinations to find the best prediction results. Hence, in the next section, we observe the prediction results based on feature learning.

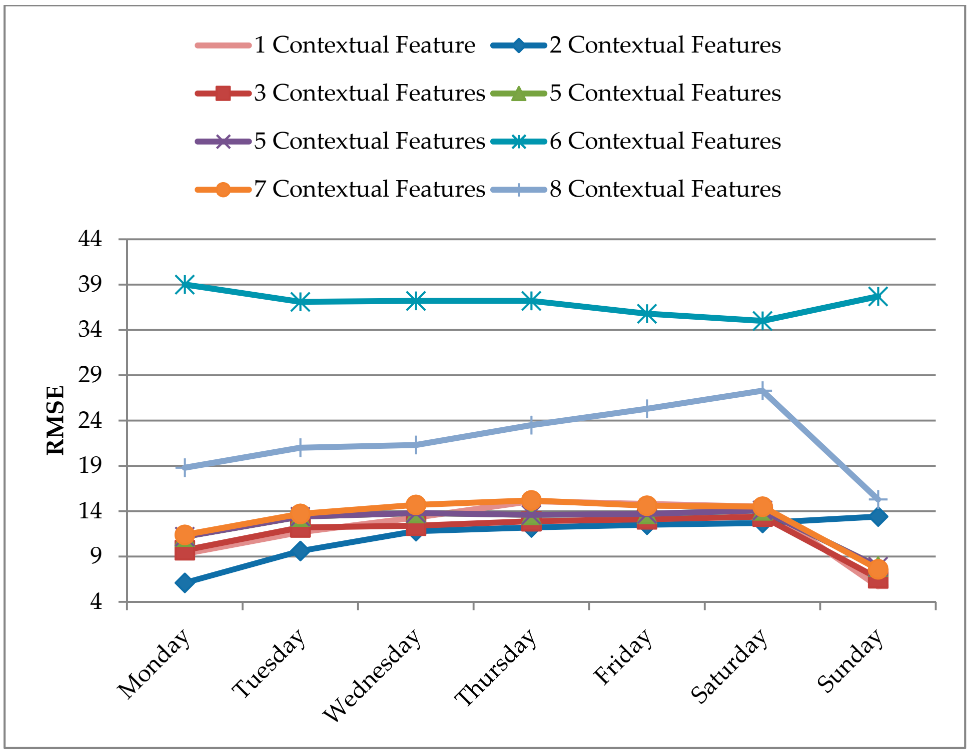

Figure 8. Root mean square error (RMSE) value for each day in the prediction of test data.

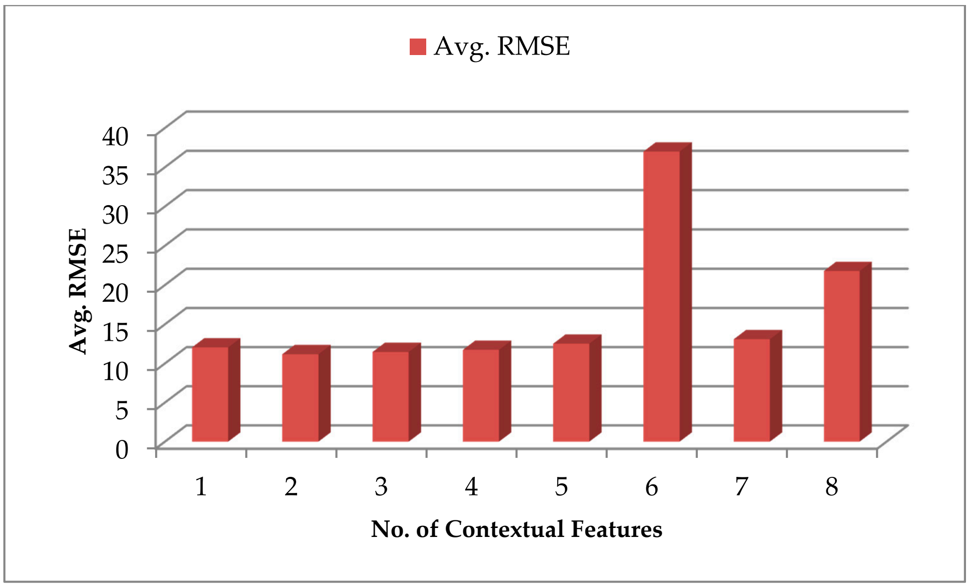

Figure 9. Avg. RMSE Score with varying number of features.

6.2. Prediction Mechanism with Feature Learning

In this subsection, we observe the prediction performance for multivariate LSTM algorithm with feature learning.

The model starts training with the entire dataset containing timestamp, energy, and 11 contextual features such as temperature, humidity, dew point, conditions, gust speed, precipitation mm, sea level pressure, visibility km, wind speed km/h, wind direction, and wind direction degrees. The model calculates feature relevance and learns feature relevance score based on features' history relevance after each cycle. An updated set of features is used for predictions in the next cycle. Figure 10 shows the prediction result plotting of predicted energy values against actual energy values. We can observe that deviations in the predicted results do not seem to be very high as compared to the previous best-achieved results with Dataset 2 in Section 6.1.

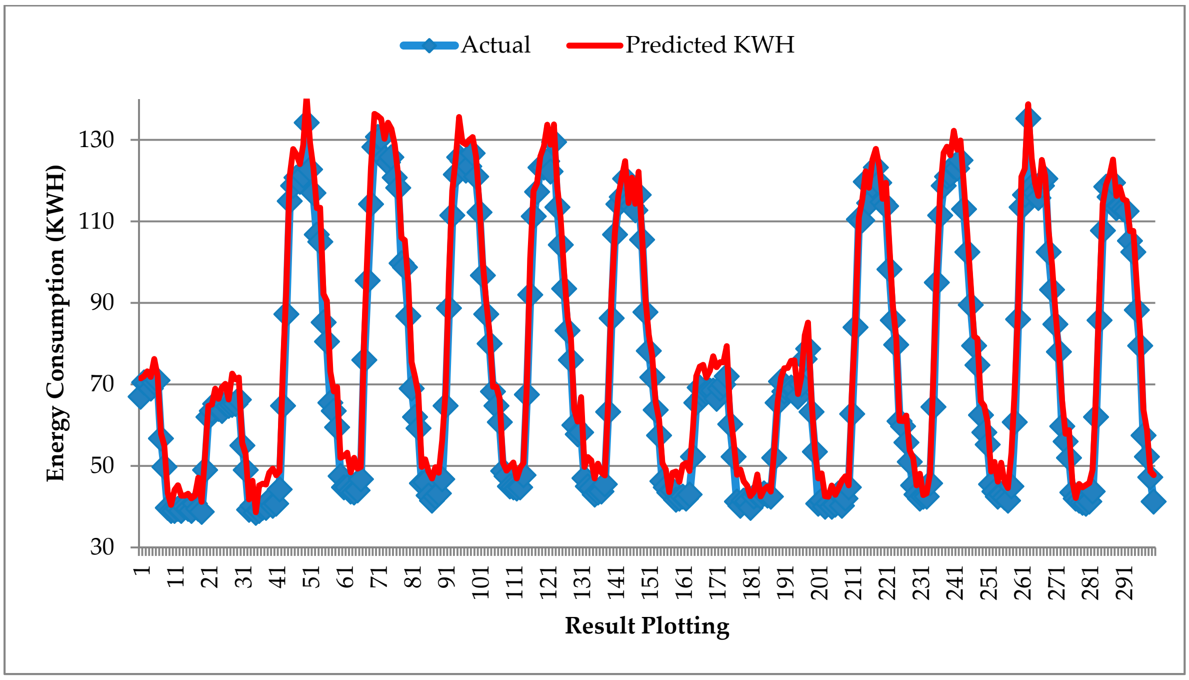

Figure 10. Plotting of predicted values against actual values with feature learning.

Figure 11 shows the RMSE score for the prediction results for LSTM with feature learning. The feature learning module updates the dataset based on calculated feature relevance. These prediction results were obtained after four cycles of LSTM algorithm training with feature learning adaptations. The average RMSE obtained was 4.70. We could observe that feature learning learns the most relevant feature very quickly, and also, the error between predicted values in contrast to actual values was reduced drastically.

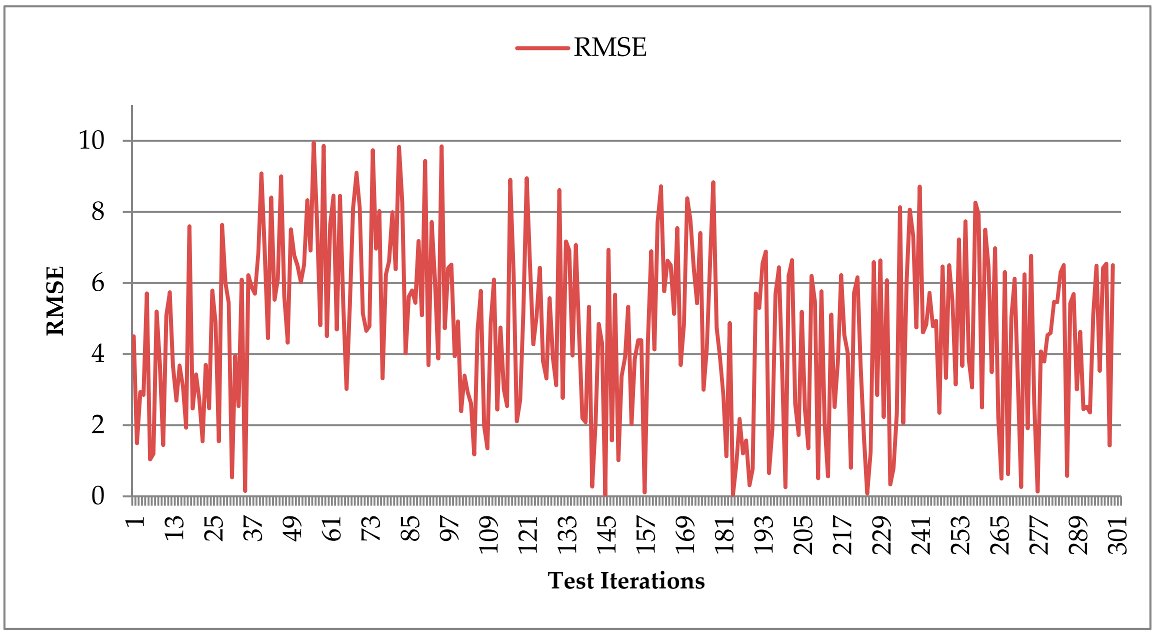

Figure 11. RMSE Score for Predictions with Feature Learning.

If we consider finding the best combination of features without the feature learning approach, then we have to check for all the possible combinations of features. If we consider our scenario of 11 contextual features, then the possible number of contextual features' combinations will be 2047. If we consider the self-selected contextual features, then the possible number of contextual features' combinations will be 255. Even in the case of eight contextual features, the total number of training cycles to find the best possible prediction results will be costly at processing powers and more time-consuming. In the table below, we present the comparison analysis for prediction model results for running of possible combinations for X = 8 and X = 11 with the LSTM model; in comparison with the prediction performance results for our proposed feature learning-based adaptive LSTM approach (Table 9).

Table 9. Datasets based on contextual feature ranking.

Algorithmic Approach	Contextual Features	Running Cycles	Execution Time (Seconds)
Multivariate LSTM	X = 11	2036	581,464.23
Multivariate LSTM	X = 8	247	66,066.15
Multivariate LSTM with Feature Learning	X = 8	4	1195.27

6.3. Predictive Learning Based Optimal Control Mechanism

In this section, we examine the performance of a predictive learning-based optimal control mechanism. We perform the comparison analysis between optimal control mechanisms with predictive learning and without predictive learning.

Figure 12 shows the comparisons of the results between predictive learning-based optimization scheme and nonpredictive learning-based optimization scheme. The y-axis shows the actuators which are being controlled at the smart home, and x-axis shows the energy consumption cost for actuator control. The energy consumption cost was measured in South Korean Won (KRW). In the results, we can clearly see that energy cost was optimized and reduced when a predictive learning-based optimization mechanism was used compared to the optimization mechanism where no predictive learning was implemented.

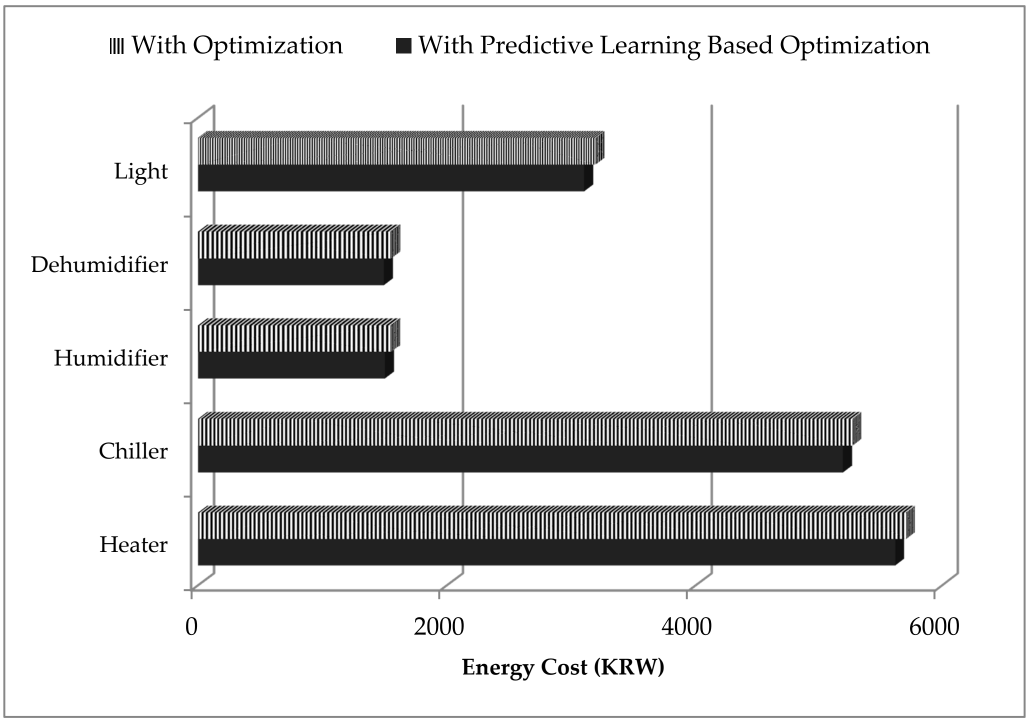

Figure 12. Energy cost comparisons for with and without predictive learning-based optimization scheme.

In Figure 13, we present the comparison of average energy consumption for optimal actuator control with and without predictive learning. In the results, we can observe that average energy consumption was reduced with predictive learning-based optimization actuator control.

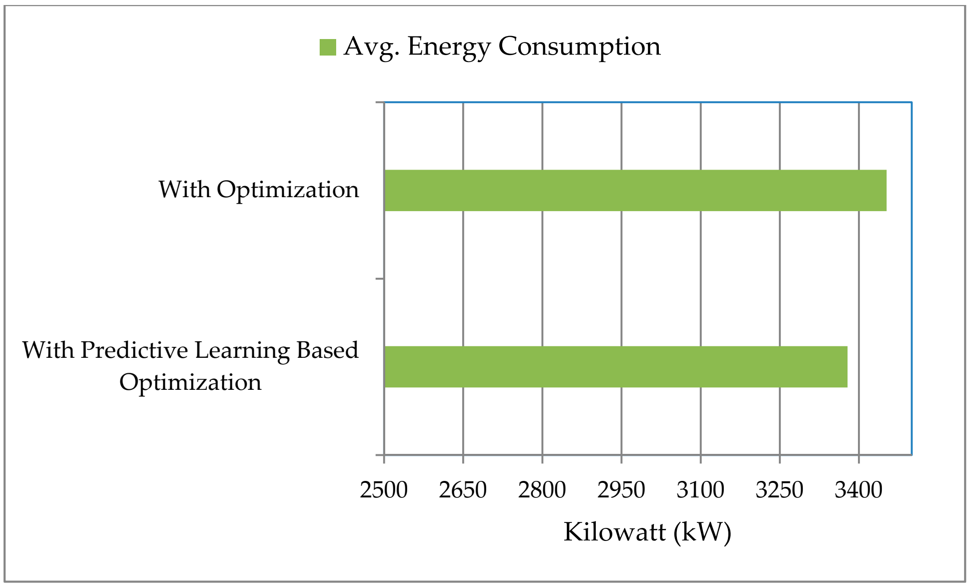

Figure 13. Average energy consumption comparisons for with and without predictive learning-based optimization scheme.

If we observe the above figures, we can see that in every actuator, we were getting improvement with our proposed predictive learning-based approach. Lights were 2.9% more cost-efficient, while dehumidifiers and humidifiers showed about 3% improvement in reducing the cost. Chiller and heater also showed about a 1.4% and 1.6% decrease in the consumption cost compared to other approaches (i.e., optimizations without predictive learning). All these comparisons are listed in Figure 12. If we consider total energy consumption of the whole system, we again see our proposed scheme outperforming the default setup. Our proposed scheme consumed approximately 3378 kilowatts, while the default system consumed 3420 kilowatts. This gave us around 2.18% improvement over the existing system. The calculation is summarized in Figure 13.

7. Conclusions

In this work, we presented a predictive learning-based optimal control mechanism for actuators deployed in smart environments. At first, we proposed a feature learning-based smart home energy prediction solution that aims to improve the prediction accuracies by filtering the contextual data based on history learnings. The algorithms used in the proposed mechanism were LSTM for energy predictions and ANN-PSO for feature learning within LSTM cycles. Our proposed prediction model takes all available features at the first cycle of energy prediction and works its way down to learn the most impactful features. The proposed model also learns to re-add features based on history learned relevance at a given prediction time. We used our implemented prediction module to further aid the optimization of a smart home actuators' control mechanism. The optimization model performance was enhanced by adding the predictive learning mechanism to it. The integration of predictive learning with optimal actuator control aims to reduce energy consumption with less dependency on the user-defined parameters for optimal control.

When analyzing the result, we compared how feature learning can dramatically increase the prediction model's performance. It finds the set of best suited contextual features with very few cycles of learning. The proposed solution can be very effective for scenarios where the dataset contains large numbers of features. It can help extract the most relevant features efficiently. Additionally, in many cases, researchers often work with datasets where they might not have complete knowledge of the data field. In such cases, the proposed model can also help researchers with feature learning and save a fair amount of time.

As the results demonstrate, our proposed system (i.e., predictive learning-based optimal control mechanism) clearly reduced the whole system's energy consumption. Saving energy is of critical importance; hence, the proposed integration can largely benefit smart homes' optimal actuator control systems.

Author Contributions: S.M. conceptualized the idea, performed data curation process, worked on methodology and wrote the original draft of the manuscript; W.S. worked on visualization, results analysis, related study comparisons, manuscript finalization reviewing and editing; K.T.L. arranged resources and funding, supervised the work, did investigation and formal analysis; D.K. was project administrator, supervised the work and proofread the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education under Grant 2020R1I1A3070744, and this research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2019M3F2A1073387).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xiao, L.; Wang, J.; Hou, R.; Wu, J. A combined model based on data pre-analysis and weight coefficients optimization for electrical load forecasting. *Energy* **2015**, *82*, 524–549. [CrossRef]
2. Li, H.-Z.; Guo, S.; Li, C.-J.; Sun, J.-Q. A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm. *Knowl. Based Syst.* **2013**, *37*, 378–387. [CrossRef]
3. Yang, Y.; Wu, J.; Chen, Y.; Li, C. A New Strategy for Short-Term Load Forecasting. *Abstr. Appl. Anal.* **2013**, *2013*, 1–9. [CrossRef]
4. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
5. Qing, X.; Niu, Y. Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM. *Energy* **2018**, *148*, 461–468. [CrossRef]
6. Yan, K.; Li, W.; Ji, Z.; Qi, M.; Du, Y. A Hybrid LSTM Neural Network for Energy Consumption Forecasting of Individual Households. *IEEE Access* **2019**, *7*, 157633–157642. [CrossRef]
7. Wang, Y.; Gan, D.; Sun, M.; Zhang, N.; Lu, Z.; Kang, C. Probabilistic individual load forecasting using pinball loss guided LSTM. *Appl. Energy* **2019**, *235*, 10–20. [CrossRef]
8. Heidari, A.; Khovalyg, D. Short-term energy use prediction of solar-assisted water heating system: Application case of combined attention-based LSTM and time-series decomposition. *Sol. Energy* **2020**, *207*, 626–639. [CrossRef]
9. Kong, W.; Dong, Z.Y.; Jia, Y.; Hill, D.J.; Xu, Y.; Zhang, Y. Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. *IEEE Trans. Smart Grid* **2019**, *10*, 841–851. [CrossRef]
10. Hrnjica, B.; Mehr, A.D. Energy demand forecasting using deep learning. In *Smart Cities Performability, Cognition, & Security*; Springer: Cham, Switzerland, 2020; pp. 71–104.
11. Wang, J.Q.; Du, Y.; Wang, J. LSTM based long-term energy consumption prediction with periodicity. *Energy* **2020**, *197*, 117197. [CrossRef]
12. Jin, Y.; Guo, H.; Wang, J.; Song, A. A Hybrid System Based on LSTM for Short-Term Power Load Forecasting. *Energies* **2020**, *13*, 6241. [CrossRef]
13. Shafqat, W.; Byun, Y.-C. A Context-Aware Location Recommendation System for Tourists Using Hierarchical LSTM Model. *Sustainability* **2020**, *12*, 4107. [CrossRef]
14. Wu, Y.; Liu, Z.; Xu, W.; Feng, J.; Palaiahnakote, S.; Lu, T. Context-Aware Attention LSTM Network for Flood Prediction. In Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 20 August 2018; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2018; pp. 1301–1306.
15. Paudel, P.; Kim, S.; Park, S.; Choi, K.-H. A Context-Aware IoT and Deep-Learning-Based Smart Classroom for Controlling Demand and Supply of Power Load. *Electronics* **2020**, *9*, 1039. [CrossRef]
16. Siegel, J.E.; Das, A.; Sun, Y.; Pratt, S.R. Safe energy savings through context-aware hot water demand prediction. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103481. [CrossRef]
17. Lee, J.M.; Hauskrecht, M. Recent context-aware lstm for clinical event time-series prediction. In Proceedings of the Conference on Artificial Intelligence in Medicine in Europe, Poznan, Poland, 26–29 June 2019; Springer: Cham, Switzerland, 2019; pp. 13–23.
18. Kaselimi, M.; Doulamis, N.; Voulodimos, A.; Protopapadakis, E.; Doulamis, A. Context Aware Energy Disaggregation Using Adaptive Bidirectional LSTM Models. *IEEE Trans. Smart Grid* **2020**, *11*, 3054–3067. [CrossRef]

19. Chen, K.; Zhang, Y.; Wang, Q.; Hu, J.; Fan, H.; He, J. Scale- and Context-Aware Convolutional Non-Intrusive Load Monitoring. *IEEE Trans. Power Syst.* **2019**, *35*, 2362–2373. [CrossRef]
20. Kim, T.-Y.; Cho, S.-B. Predicting the Household Power Consumption Using CNN-LSTM Hybrid Networks. In Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning, Madrid, Spain, 21 November 2018; Springer Science and Business Media LLC: Cham, Switzerland, 2018; pp. 481–490.
21. Li, Y.; He, Y.; Zhang, M. Prediction of Chinese energy structure based on Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM). *Energy Sci. Eng.* **2020**, *8*, 2680–2689. [CrossRef]
22. Park, K.; Yoon, S.; Hwang, E. Hybrid load forecasting for mixed-use complex based on the characteristic load de-composition by pilot signals. *IEEE Access* **2019**, *7*, 12297–12306. [CrossRef]
23. Ghimire, S.; Deo, R.C.; Raj, N.; Mi, J. Deep solar radiation forecasting with convolutional neural network and long short-term memory network algorithms. *Appl. Energy* **2019**, *253*, 113541. [CrossRef]
24. De Jesús, D.A.R.; Mandal, P.; Chakraborty, S.; Senjyu, T. Solar pv power prediction using a new approach based on hybrid deep neural network. In Proceedings of the 2019 IEEE Power & Energy Society General Meeting (PESGM), Atlanta, GA, USA, 4 August 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–5.
25. Li, P.; Zhou, K.; Lu, X.; Yang, S. A hybrid deep learning model for short-term PV power forecasting. *Appl. Energy* **2020**, *259*, 114216. [CrossRef]
26. Bouktif, S.; Fiaz, A.; Ouni, A.; Serhani, M.A. Optimal Deep Learning LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches. *Energies* **2018**, *11*, 1636. [CrossRef]
27. Abdollahi, M.; Khaleghi, T.; Yang, K. An integrated feature learning approach using deep learning for travel time prediction. *Expert Syst. Appl.* **2020**, *139*, 112864. [CrossRef]
28. Sun, G.; Jiang, C.; Wang, X.; Yang, X. Short-term building load forecast based on a data-mining feature selection and LSTM-RNN method. *IEEJ Trans. Electr. Electronic Eng.* **2020**, *15*, 1002–1010. [CrossRef]
29. Shah, A.S.; Nasir, H.; Fayaz, M.; Lajis, A.; Ullah, I.; Shah, A. Dynamic User Preference Parameters Selection and Energy Consumption Optimization for Smart Homes Using Deep Extreme Learning Machine and Bat Algorithm. *IEEE Access* **2020**, *8*, 204744–204762. [CrossRef]
30. Rodriguez-Mier, P.; Mucientes, M.; Bugarín, A. Feature Selection and Evolutionary Rule Learning for Big Data in Smart Building Energy Management. *Cogn. Comput.* **2019**, *11*, 418–433. [CrossRef]
31. Gonzalez-Vidal, A.; Jimenez, F.; Gomez-Skarmeta, A.F. A methodology for energy multivariate time series fore-casting in smart buildings based on feature selection. *Energy Build.* **2019**, *196*, 71–82. [CrossRef]
32. Wang, R.; Lu, S.; Feng, W. A novel improved model for building energy consumption prediction based on model integration. *Appl. Energy* **2020**, *262*, 114561. [CrossRef]
33. Jallal, M.A.; González-Vidal, A.; Skarmeta, A.F.; Chabaa, S.; Zeroual, A. A hybrid neuro-fuzzy inference system-based algorithm for time series forecasting applied to energy consumption prediction. *Appl. Energy* **2020**, *268*, 114977. [CrossRef]
34. Tian, C.; Ma, J.; Zhang, C.; Zhan, P. A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network. *Energies* **2018**, *11*, 3493. [CrossRef]
35. Malik, S.; Kim, D. Prediction-Learning Algorithm for Efficient Energy Consumption in Smart Buildings Based on Particle Regeneration and Velocity Boost in Particle Swarm Optimization Neural Networks. *Energies* **2018**, *11*, 1289. [CrossRef]
36. Malik, S.; Lee, K.; Kim, D. Optimal Control Based on Scheduling for Comfortable Smart Home Environment. *IEEE Access* **2020**, *8*, 218245–218256. [CrossRef]
37. Available online: https://www.kaggle.com/claytonmiller/building-data-genome-project-v1 (accessed on 15 January 2021).