International Journal of Analysis and Applications

Optimizing AWS Cloud Resource Management: Predicting EC2 Instance CPU Utilization using LSTM and ARIMA Models

Hanita Daud[1], Nazleeni Samiha Haron[2], Nor Farisha M Krishnan[3], Sieow Yeek Tan[4], Mohd Amirul Faiz Abdul Wahab[4], Ahmad Amirul Adlan Azhar[4], Anis Zahirah Zubir[1], Sofea Balqis Sri Emirlee[1], Diaa S. Metwally[5], Ahmad Abubakar Suleiman[6,*]

[1]*Department of Fundamental and Applied Sciences, Universiti Teknologi Petronas, Seri Iskandar, Perak 32610, Malaysia*

[2]*Department of Computer and Information Sciences, Universiti Teknologi Petronas, Seri Iskandar, Perak 32610, Malaysia*

[3]*Centre for Research in Data Science (CeRDaS), Universiti Teknologi PETRONAS, Seri Iskandar, Perak 32610, Malaysia*

[4]*Exchange Square, Bursa Malaysia Berhad, Bukit Kewangan, 50200 Kuala Lumpur, Malaysia*

[5]*Deanship of Scientific Research, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia*

[6]*Department of Statistics, Aliko Dangote University of Science and Technology, Wudil 713281, Nigeria*

**Corresponding author:* ahmadabubakar31@gmail.com

ABSTRACT. Amazon Web Services (AWS) Elastic Compute Cloud (EC2) instances offer scalable computing resources crucial for various applications. Accurate prediction of CPU utilization is essential for efficient resource management and cost optimization in cloud environments. This study investigates the performance of machine learning models, specifically Long Short-Term Memory (LSTM) networks and AutoRegressive Integrated Moving Average (ARIMA) models, for forecasting CPU utilization of AWS EC2 instances in both development and production environments. By employing historical data from both environments, the research aims to extend predictive horizons and improve forecasting accuracy. We evaluate and compare model performance using Mean Squared Error (MSE) and fitting times. Results reveal that ARIMA models consistently outperform LSTM models in terms of MSE and computational

Received Jul. 25, 2025

2020 Mathematics Subject Classification. 62M10.

Key words and phrases. AWS EC2 instances; CPU utilization; cloud resource management; machine learning models; ARIMA; LSTM; forecasting.

https://doi.org/10.28924/2291-8639-23-2025-261

ISSN: 2291-8639

efficiency, demonstrating superior performance in both environments. LSTM models, despite their potential, show higher variability and longer fitting times, especially with hyperparameter tuning. This paper highlights the critical role of model selection and tuning in enhancing forecasting capabilities and operational efficiency in cloud resource management. The findings contribute valuable insights for optimizing resource allocation and cost management in AWS cloud services.

1. Introduction

Cloud computing has transformed how businesses and individuals access computing power, storage, and services over the internet. Amazon Web Services (AWS) is a leading cloud platform that provides a wide range of on-demand services, including artificial intelligence, machine learning, and the Internet of Things (IoT) [1]. Alongside other major providers such as Google Cloud, Microsoft Azure, IBM Cloud, and Alibaba Cloud, AWS enables scalable and cost-effective deployment of applications worldwide [2]. Cloud storage is increasingly popular for both individuals and businesses seeking efficient off-site data backup solutions [3, 4].

Research on cloud computing has evolved significantly, with early studies focusing on virtualization and distributed computing [5]. The concept gained widespread adoption in the early 2000s, and the launch of AWS in 2006 marked a turning point in the commercial viability of cloud services. Since then, research has explored various aspects, including performance optimization, scalability, and security [6, 7]. The shift toward cloud-based solutions has enabled organizations across industries to access extensive computing resources, often beyond what is feasible with local infrastructure [8-10]. However, effective management of these services requires specialized skills and precise resource analysis to avoid inefficiencies and excessive costs [11-14].

Despite advancements, cloud resource management still presents challenges, especially in large-scale systems [15, 16]. Proper allocation of computational resources is essential to maintain performance and control expenses. AWS Elastic Compute Cloud (EC2) instances, which offer scalable computing capacity, require continuous monitoring and optimization. One key aspect is predicting CPU utilization, as accurate forecasts help prevent under-provisioning (which degrades performance) and over-provisioning (which increases costs) [17, 18]. Previous studies have shown that predicting CPU utilization is challenging [19]. While recurrent neural networks (RNNs) provide accurate short-term predictions, longer forecasts up to 30 minutes are necessary for proactive capacity adjustments [20, 21].

To address this, our research employs Long Short-Term Memory (LSTM) networks, known for their superior performance in time series forecasting. Additionally, we apply the AutoRegressive Integrated Moving Average (ARIMA) model to further enhance predictive accuracy. By analyzing historical data, these models help optimize AWS resource allocation and improve cost efficiency [20, 22, 23]. Our study also incorporates anomaly detection techniques

using Random Forest (RF) and Artificial Neural Networks (ANN) to identify inefficiencies in resource utilization.

The novelty of our work lies in extending the predictive horizon for CPU utilization while improving accuracy. By leveraging established machine learning techniques, we provide practical insights for proactive resource management. This approach not only enhances AWS operational efficiency but also contributes to sustainable cloud computing practices.

The primary motivations of this research are summarized as follows:

i. To independently evaluate and compare the forecasting capabilities of ARIMA and LSTM models for CPU utilization prediction on AWS EC2 instances.
ii. To investigate the impact of hyperparameter tuning on the performance of ARIMA and LSTM models, specifically focusing on optimizing prediction accuracy for CPU utilization in EC2 instances.
iii. To implement data pre-processing techniques tailored for time series data, enhancing model performance and prediction reliability.
iv. To conduct exploratory data analysis (EDA) to identify patterns, trends, and anomalies in the dataset, providing valuable insights for model selection and refinement.

The structure of this paper is as follows: Section 2 provides a comprehensive literature review. Section 3 details the research methodology applied in this research, including the ARIMA and LSTM models, along with the mathematical formulations. Section 4 provides the results and discussion on the application of LSTM and ARIMA models for CPU utilization prediction. Finally, Section 5 concludes the paper with key findings and future directions.

2. Literature Review

This section reviews studies on CPU utilization prediction in cloud computing, particularly on AWS. Accurate forecasting is crucial for optimizing resource allocation and reducing costs. Borra [24] describes cloud computing as a flexible IT resource model, while Balaji, et al. [25] emphasizes user-centric pricing strategies. AWS, as noted by Kaufman [26], provides scalable, cost-effective cloud services that enhance operational efficiency, enabling global deployment [27].

Several studies compare predictive models for CPU utilization. Preetham, et al. [28] found that LSTM models outperform ARIMA in handling complex cloud workloads, improving resource management, and reducing SLA violations. Similarly, Nguyen, et al. [29] demonstrated that LSTM-based models significantly enhance prediction accuracy and adaptability compared to ARIMA. Osypanka and Nawrocki [30] proposed a cost-optimization approach combining machine learning, anomaly detection, and particle swarm optimization, achieving an 85% cost

reduction in Azure environments. Duggan, et al. [20] explored RNNs for CPU forecasting, showing superior performance in dynamic cloud environments compared to traditional methods.

These studies collectively highlight the importance of machine learning techniques in optimizing cloud resource management, reducing costs, and improving operational efficiency.

3. Methodology

This section describes the approaches used to create and assess the time series forecasting models known as LSTM and ARIMA. Data collection, preprocessing, model creation, assessment measures, and implementation tools are all included in the methodology.

3.1. Dataset

The dataset for this study was sourced from Bursa Malaysia Berhad, a reliable and comprehensive data provider. However, due to confidentiality, the data is not publicly available. It consists of CPU utilization metrics from Amazon EC2 instances, collected from two distinct environments: production and development. The production environment comprises 11 instances, spanning data from December 3, 2022, to March 1, 2024, while the development environment has 10 instances, covering the period from January 1, 2023, to November 30, 2023. These instances run on various operating systems, including Windows and Red Hat Enterprise Linux (RHEL). To ensure data integrity, a thorough preprocessing step was performed to address any missing values, enhancing the accuracy and reliability of the analysis.

3.3.1. Data Preprocessing

Data preprocessing ensures reliability and accuracy before model training or analysis. In this study, both production and development datasets contained missing values, representing periods of CPU inactivity. Instances with more than 50% missing data were excluded to maintain data integrity. For those with fewer missing values, mean imputation was applied, replacing missing values with the average of the observed data for that instance. This approach preserves dataset consistency without significantly altering its statistical properties.

3.2. Time Series Forecasting

Forecasting is a critical process for businesses and governments, as it helps in developing future strategies based on scientifically calculated predictions. An effective forecast should be accurate, reliable, timely, easy to understand, cost-efficient, and as simple as possible [31]. Forecasting problems are generally categorized into short-term, medium-term, and long-term forecasts [32]. Short-term forecasts predict events over a few days, weeks, or months, while medium-term forecasts extend up to two years and are often used for operations management and budgeting [33]. Long-term forecasts, which span several years, are typically used for strategic planning.

Short- and medium-term forecasting techniques focus on identifying patterns in historical data to make reliable predictions [34, 35]. This study explores ARIMA and LSTM models for stock

market movement prediction, as they are robust univariate forecasting methods capable of projecting future time series values.

3.3. ARIMA Model

The ARIMA model, also known as the Box-Jenkins model, is a widely used approach for time series forecasting. It consists of three components: Auto-Regressive (AR), which models a variable based on its past values; Integrated (I), which ensures stationarity by differencing observations; and Moving Average (MA), which captures dependencies between an observation and past errors [36, 37].

Since most real-world time series data are non-stationary, ARIMA first ensures stationarity using the Augmented Dickey-Fuller (ADF) test before applying the model [37, 38]. It then converts non-stationary data into a stationary form through differencing. The Box-Jenkins approach, a variation of ARIMA, analyzes differences between consecutive values rather than absolute values, making it particularly useful for forecasting financial markets [36]. Figure 1 shows the overall structure of Box-Jenkins. The figure highlights three important processes.

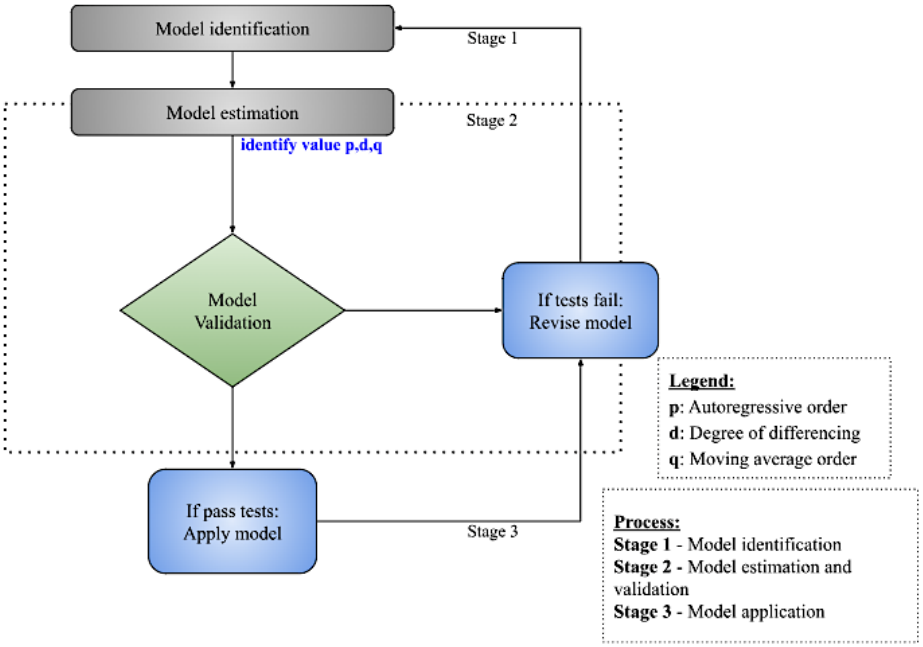


FIGURE 1. Box Jenkins structure.

The modeling process consists of two key stages: model identification and model estimation, as well as validation. If validation tests fail, the model is revised and reassessed; if they pass, the model is applied.

In this study, the Augmented Dickey-Fuller (ADF) test serves as a crucial diagnostic tool for evaluating the stationarity of time series data. The decision to rely on p-values from the ADF test is based on its ability to detect unit roots, which indicate non-stationarity. This ensures that trends or spurious relationships are identified, improving the reliability of the forecasting models.

For model selection, the auto.arima function in Python automates the determination of the (p, d, q) parameters of the ARIMA model, where:

i. 'p' is the number of lag observations included in the model (autoregressive part).
ii. 'd' is the number of times that the raw observations are different (integrated part).
iii. 'q' is the size of the moving average window.

Once the parameters were determined, the ARIMA model was fitted to the training data. The residuals of the fitted model were analyzed to ensure no significant patterns were left unmodeled and to confirm the model's adequacy. The Ljung-Box test was used to verify the white noise characteristics of the residuals, ensuring the model's validity. A comprehensive flow chart detailing the ARIMA model approach for time series forecasting is presented in Figure 2. The ARIMA model can be represented by the following equation (1):

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + ... + \theta_q \varepsilon_{t-q} + \varepsilon_t, \quad (1)$$

where

a. y_t is the value at time t.
b. c is a constant term.
c. $\phi_1, \phi_2, ..., \phi_p$ are the coefficients of the autoregressive part.
d. $\theta_1, \theta_2, ..., \theta_q$ are the coefficients of the moving average part.
e. ε_t is the error term at time (white noise).

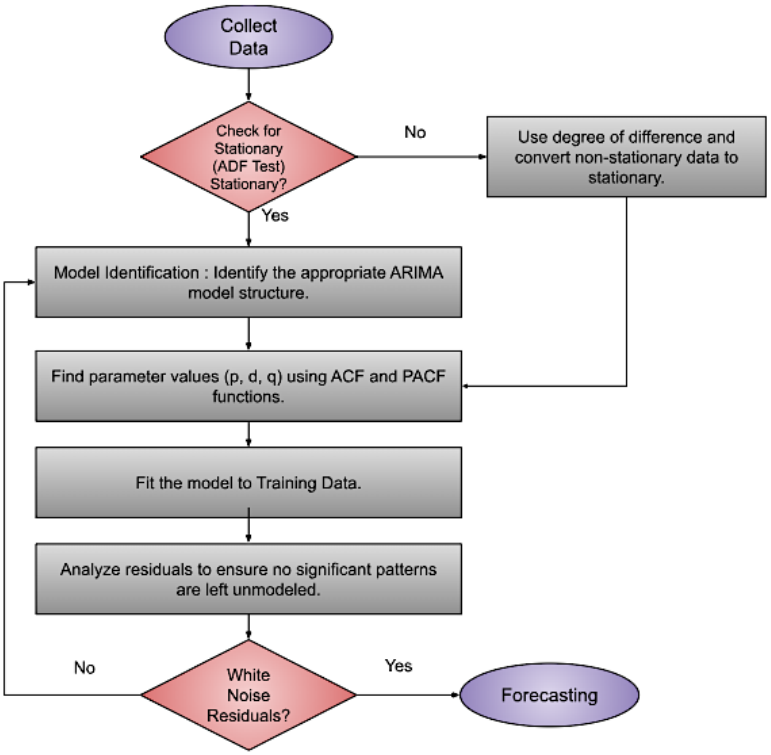


FIGURE 2. The approach followed for forecasting using ARIMA.

3.4. LSTM Model

Artificial neural networks (ANNs) are computational models inspired by biological neural networks, consisting of interconnected units called neurons organized in layers. A specific type of ANN, known as Recurrent Neural Networks (RNNs), is designed to process sequences of data where each value depends on previous ones. Long Short-Term Memory (LSTM) networks, a specialized form of RNNs, excel at remembering information over long periods [39]. These models are particularly effective for predicting, processing, and classifying time series data [40]. LSTM networks stand out for their ability to handle time-dependent data, thanks to their four interacting layers that communicate uniquely during data processing. The model's structure often takes the form of a chain. An analysis block diagram of the method used in our study to predict CPU utilization is shown in Figure 3.

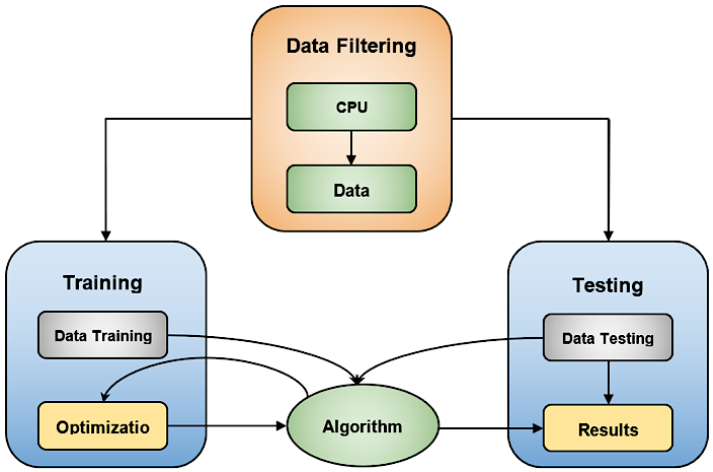


FIGURE 3. The approach followed for forecasting using LSTM.

The main feature of LSTM networks is their ability to store and process information over long sequences. Each LSTM unit has four parts that work together, as shown in Figure 4.

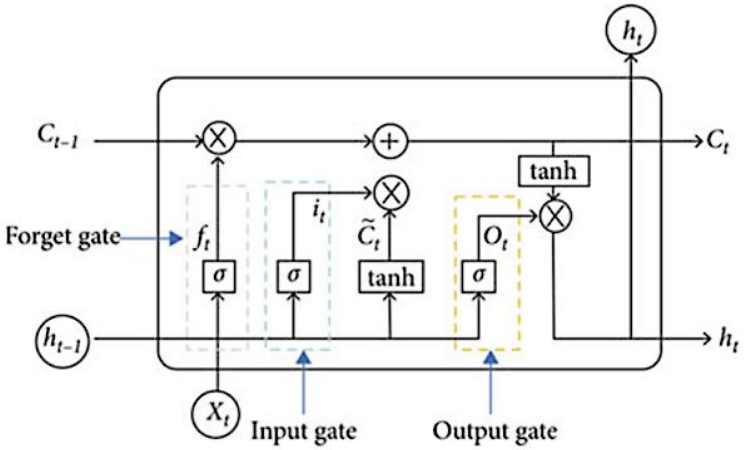


FIGURE 4. Structure of the RNN-LSTM algorithm

The line running through these units is called the cell state, which carries information through the LSTM network, updating it as needed. LSTM networks have three types of gates: the input gate, the forget gate, and the output gate. The process of identifying and excluding data in LSTM is controlled by the forget gate. The forget gate takes the output of the last LSTM unit h_{t-1} at time $t-1$ and the current input X_t at time t. The sigmoid function also determines which parts of the previous output should be discarded, producing a vector f_t with values ranging from 0 to 1 for each element in the cell state C_{t-1} (Figure 4). If the output is 1, the information is retained, and if it is 0, the information will be discarded. The equation for the forget gate is :

$$f_t = \sigma\left(W_f.\left[h_{t-1}, X_t\right] + b_f\right), \tag{2}$$

where:

a. W_f represents the weight matrix associated with the forget gate.

b. $\left[h_{t-1}, X_t\right]$ denotes the concatenation of the current input and the previous hidden state.

c. b_f is the bias with the forget gate.

d. σ is the sigmoid activation function.

The input gate adds useful information to the cell state. First, the input is regulated using the sigmoid function to filter the values to be remembered, using the inputs h_{t-1} and X_t. Then, a vector is created using a $\tanh$ function, outputting values from -1 to 1, representing all possible values from h_{t-1} and X_t. Finally, the values of the vector and the regulated values are multiplied to obtain the useful information. The equations for the input gate are:

$$i_t = \sigma\left(W_i.\left[h_{t-1}, X_i\right] + b_i\right). \tag{3}$$

$$\widehat{C}_t = \tanh\left(W_c.\left[h_{t-1}, X_t\right] + b_c\right). \tag{4}$$

The previous state is multiplied by f_t, disregarding the information chosen to be ignored. The updated candidate values i_t. $\widehat{C}_t$ are then added, resulting in:

$$C_t = f_t.C_{t-1} + i_t.\widehat{C}_t. \tag{5}$$

From the current cell state, the output gate collects relevant information to display as output. The $\tanh$ function is first used to build a vector. After that, the data is controlled by the sigmoid function, which filters the values that need to be retained using the inputs h_{t-1} and X_t. To send the values of the vector and the controlled values as output and input to the following cell, they are finally multiplied. The equation for the output gate is:

$$O_t = \sigma\left(W_o.\left[h_{t-1}, X_t\right] + b_o\right). \tag{6}$$

These gates are essential for how LSTMs work. The input gate allows the network to learn new patterns, the forget gate prevents old, irrelevant information from cluttering the cell state, and the output gate ensures accurate predictions. By managing the flow of information with these gates, LSTMs can capture long-term dependencies in data. This makes them useful for tasks like time series forecasting and natural language processing, where understanding long-term patterns is important.

To set up the LSTM model, the data must be transformed into sequences, each of which is used to predict the next value. The dataset was split into training and testing subsets, as shown in Figure 3. An optimization process was conducted to evaluate the results, improve performance, and enhance accuracy by minimizing errors in the final forecasting. In the LSTM cell, the weight matrices and biases for the sigmoid function are crucial components for the forget gate. This gate determines and stores the input data from the new information X_t in the cell state and updates it accordingly. The sigmoid layer then decides whether the new data should be retained or discarded (0 or 1), while the $\tanh$ function assigns weights to the values based on the importance (-1 to 1). The values are multiplied to update the new cell state, which is then combined with the previous cell state C_{t-1} to form the new cell state C_t. Figure 5 illustrates the neuron processing mechanism of the LSTM model [41, 42].

In this study, the LSTM was designed with two layers, each with 4 and 32 LSTM units, to capture long-term dependencies in the data. To avoid overfitting, dropout layers with a rate of 0.0 - 0.2 were added. To assess the effectiveness of LSTM models, three different approaches were compared: using hyperparameter tuning methods such as Grid Search and Random Search, and a baseline model without hyperparameter tuning. Additionally, the training time for each model was recorded to assess the computational efficiency of each approach.

The next step involves the cell states C_{t-1} and C_t at times $t-1$ and t, respectively. In the final step, the value of h_t is determined based on the output cell state O_t. A sigmoid layer decides which parts of the cell state will contribute to the output. The output of the sigmoid gate O_t is then multiplied by the new values created by the $\tanh$ layer from the cell state C_t, with values ranging between -1 and 1.

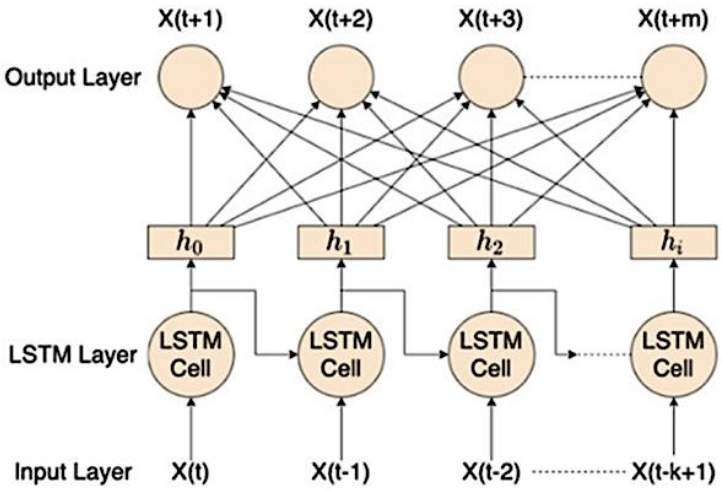


FIGURE 5. Internal LSTM model process.

To ensure the reliability and accuracy of the models, a rolling forecasting origin approach was used for model validation, evaluating their performance across various segments of the dataset. This method helps confirm that the models can make accurate predictions in different scenarios. The ARIMA and LSTM models were compared using performance metrics to identify the best forecasting model. The evaluation focused on Mean Squared Error (MSE) and training time. MSE was chosen as the key evaluation metric, as it measures the average squared error between predicted and actual values, offering a comprehensive assessment of model accuracy. A smaller MSE indicates better model performance [43].

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2, \tag{7}$$

where N is the number of observations, y_i is the actual value and $\bar{y}$ is the predicted value.

For the LSTM model, we performed three categories of analysis:

1. Using default hyperparameters

This study introduces a hyperparameter optimization algorithm using a Multilayer Perceptron (MLP) model to predict and refine hyperparameters, improving accuracy while reducing training data. Innovations include the MLP Prediction Model for better efficiency, Neighboring Value Perturbation inspired by genetic algorithms for exploring optimal solutions, and a Stability Model to ensure precision and stability across multiple runs.

2. Using hyperparameters with grid search

Grid search is a powerful hyperparameter optimization technique for fine-tuning LSTM models, improving their performance in time series forecasting. By systematically exploring hyperparameter combinations like the number of layers, units, and learning rates, grid search

enhances the LSTM's ability to learn long-term dependencies and improve accuracy in forecasting tasks [44].

3. Using hyperparameters with random search

Random search is more efficient than grid search for hyperparameter tuning in LSTM models, requiring less computational effort while still finding optimal configurations. It can explore a wider range of hyperparameter combinations, achieving similar or better performance on various datasets. This efficiency, supported by empirical evidence, makes random search an excellent baseline for optimizing LSTM models without the exhaustive search needed by grid search [45].

4. Results and Discussion

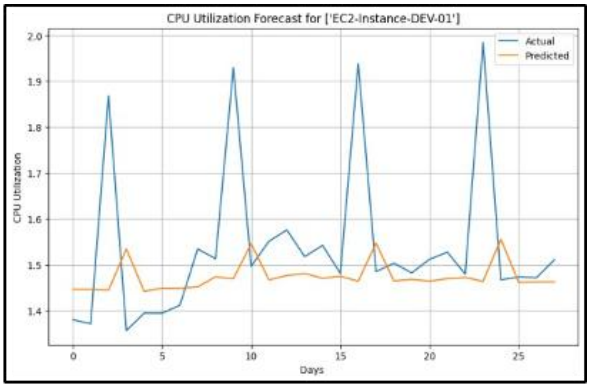
4.1. LSTM Model Analysis

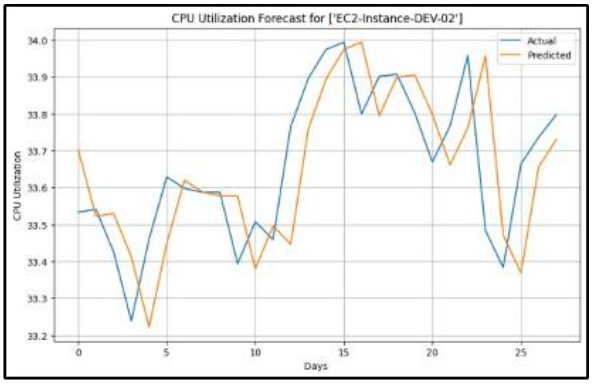
4.1.1. Default Hyperparameters

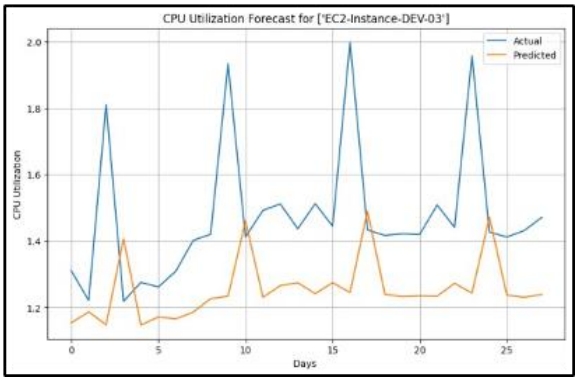
Table 1 displays the results of LSTM models with default hyperparameters in the development environment, showing varied performances across different instances. The MSE values reveal significant differences, indicating that the default parameters may not be optimal for all datasets. Training times also vary, highlighting the computational demands of LSTM models. Notably, EC2-Instance-DEV-04 achieves the lowest MSE (0.0048) with a moderate training time of 4.3 seconds, suggesting a good balance of performance and efficiency. Figure 6 illustrates the predictive performance of these models across instances in the development environment.

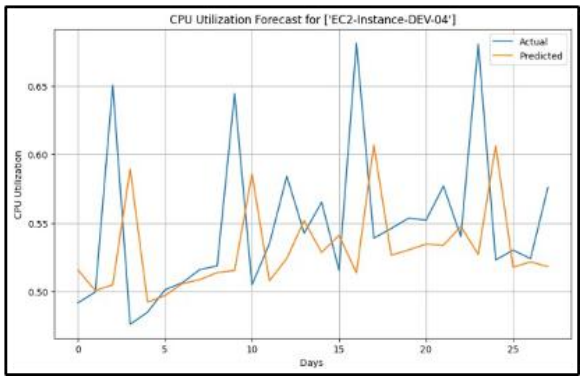
TABLE 1. Results of the LSTM models with default hyperparameters for the development environment

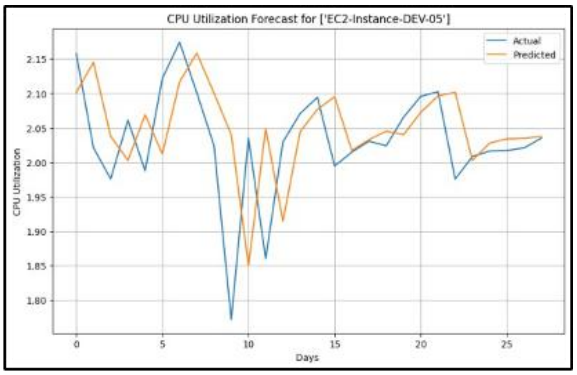
Instance ID	MSE	Training Time (s)
EC2-Instance-DEV-01	0.0354	3.5
EC2-Instance-DEV-02	0.0285	5.5
EC2-Instance-DEV-03	0.0998	4.3
EC2-Instance-DEV-04	0.0048	4.3
EC2-Instance-DEV-05	0.0086	16.4
EC2-Instance-DEV-06	0.0050	5.4
EC2-Instance-DEV-07	0.4234	3.3
EC2-Instance-DEV-08	0.0884	4.7
EC2-Instance-DEV-09	3.2938	5.4

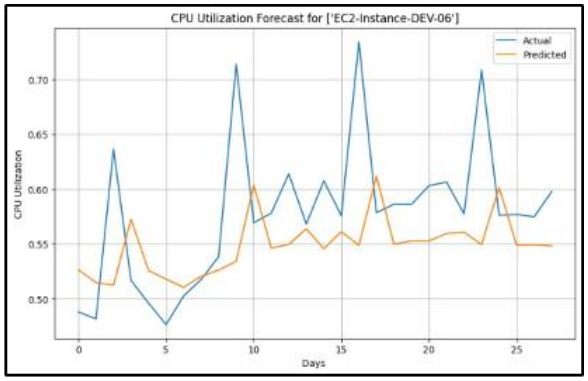

CPU Utilization Forecast for ['EC2-Instance-DEV-01']
Actual
Predicted
CPU Utilization
Days

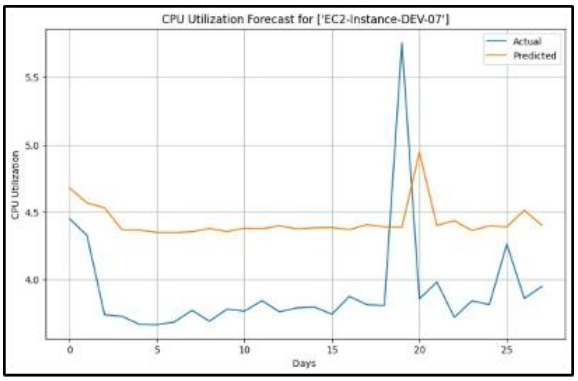

CPU Utilization Forecast for ['EC2-Instance-DEV-02']
Actual
Predicted
CPU Utilization
Days

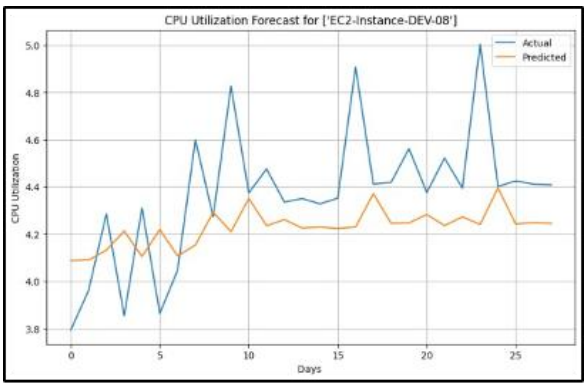

CPU Utilization Forecast for ['EC2-Instance-DEV-03']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-DEV-04']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-DEV-05']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-DEV-06']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-DEV-07']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-DEV-08']
Actual
Predicted
CPU Utilization
Days

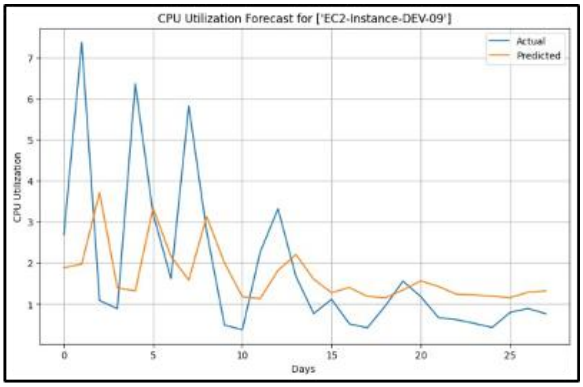
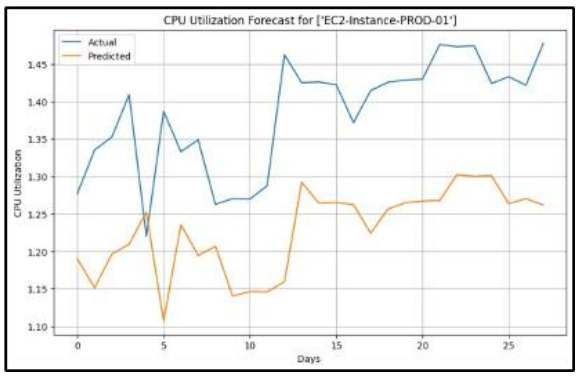


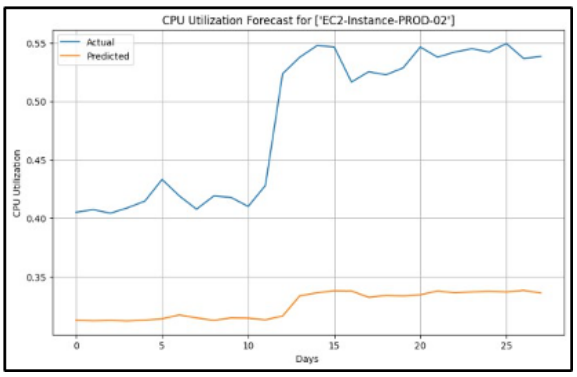
FIGURE 6. Predictive performance of the LSTM models with default hyperparameters for the development environment.

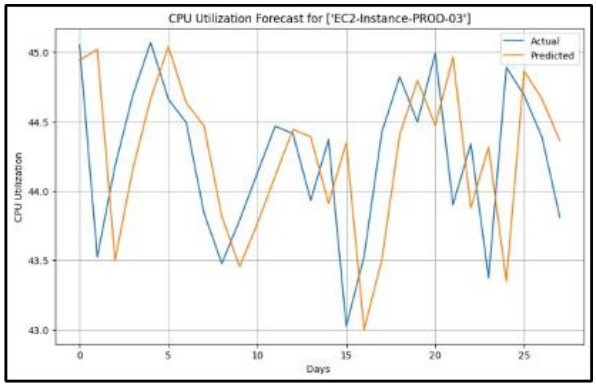
Table 2 shows the results of the LSTM models with default hyperparameters for the production environment. The result also exhibits variability in MSE and training times. The EC2-Instance-PROD-07 stands out with an MSE of 0.0008 and a training time of 7.3 seconds, indicating excellent performance. However, EC2-Instance-PROD-03 has a higher MSE of 0.4672, suggesting that the LSTM model may struggle with certain datasets in the production environment. Figure 7 displays the predictive performance of the LSTM models with default hyperparameter settings for the production environment across various instances.

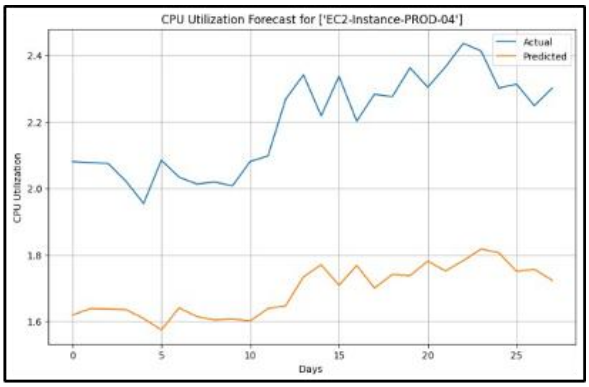
TABLE 2. Results of the LSTM models with default hyperparameters for the production environment

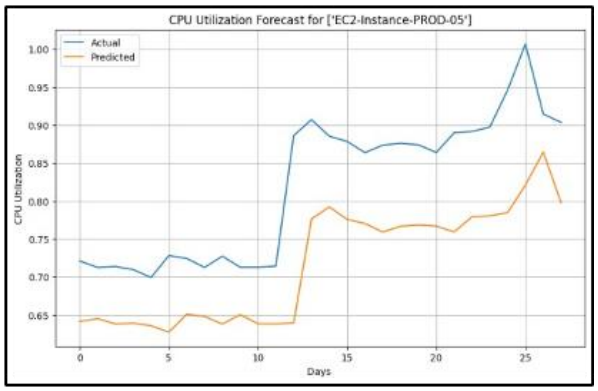
Instance ID	MSE	Training Time (s)
EC2-Instance-PROD-01	0.0278	10.1
EC2-Instance-PROD-02	0.0278	6.5
EC2-Instance-PROD-03	0.4672	6.8
EC2-Instance-PROD-04	0.2613	5.6
EC2-Instance-PROD-05	0.0120	9.7
EC2-Instance-PROD-06	0.2639	4.4
EC2-Instance-PROD-07	0.0008	7.3
EC2-Instance-PROD-08	0.0005	7.9
EC2-Instance-PROD-09	0.1053	4.1

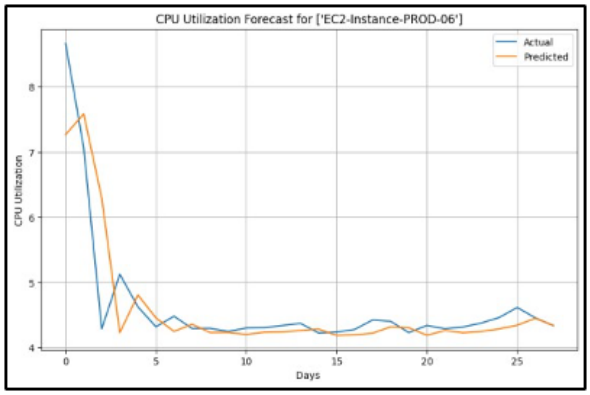

CPU Utilization Forecast for ['EC2-Instance-PROD-01']
Actual
Predicted
CPU Utilization
Days

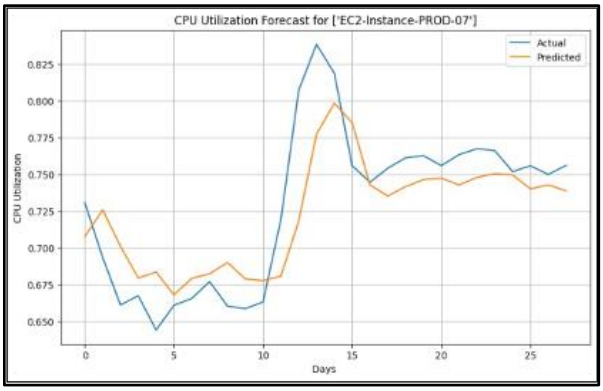

CPU Utilization Forecast for ['EC2-Instance-PROD-02']
Actual
Predicted
CPU Utilization
Days

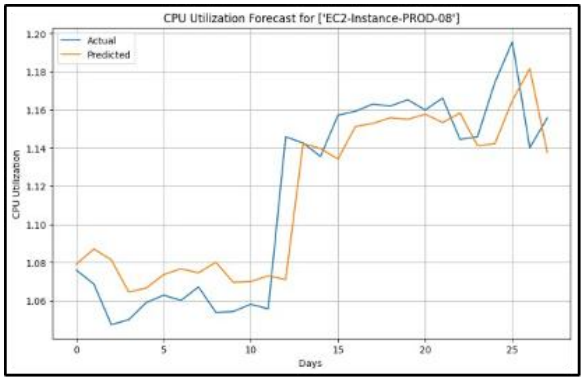

CPU Utilization Forecast for ['EC2-Instance-PROD-03']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-PROD-04']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-PROD-05']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-PROD-06']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-PROD-07']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-PROD-08']
Actual
Predicted
CPU Utilization
Days

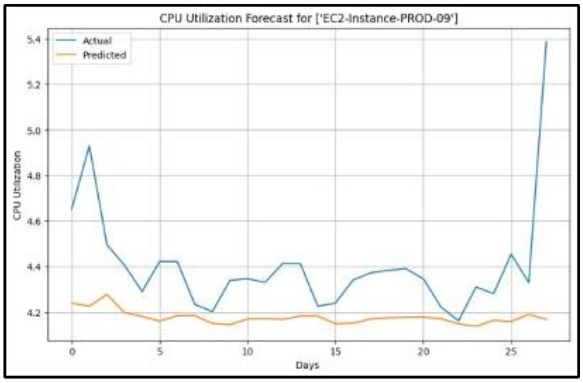


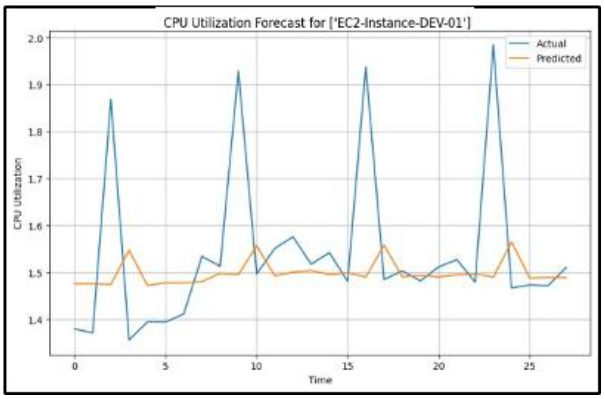
FIGURE 7. Predictive performance of the LSTM models with default hyperparameters for the production environment.

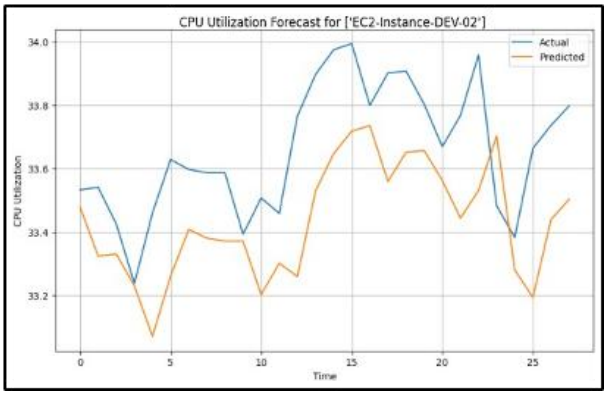
4.1.2 Grid Search Hyperparameters

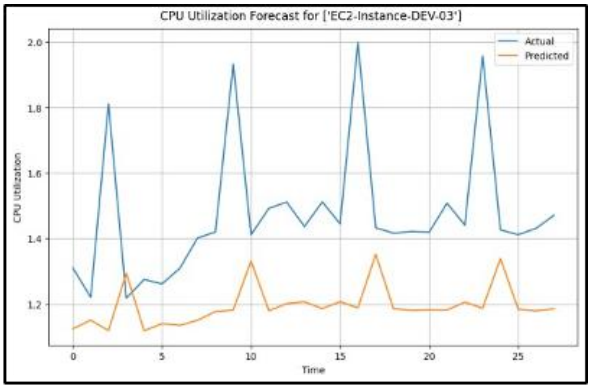
Table 3 illustrates the results of the grid search approach for the development environment, which optimizes hyperparameters, generally leads to improved MSE values compared to the default settings, albeit at the cost of increased training times. EC2-Instance-DEV-04 retains a low MSE of 0.0048, with a training time extending to 5 minutes and 39.9 seconds. This suggests that the hyperparameter tuning process was effective in maintaining performance while extending the training duration. Figure 8 portrays the predictive performance of the LSTM models with grid search hyperparameter settings for the development environment across various instances.

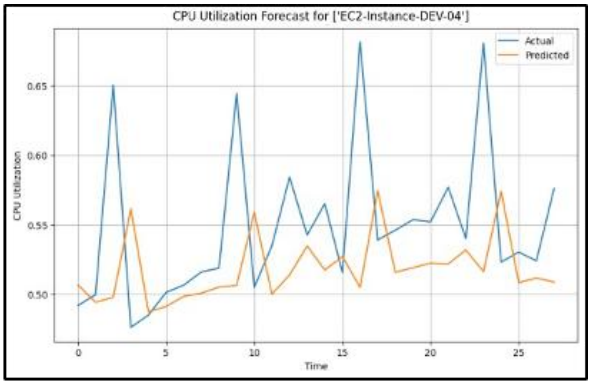
TABLE 3. Results of the LSTM models with grid search hyperparameters for the development environment

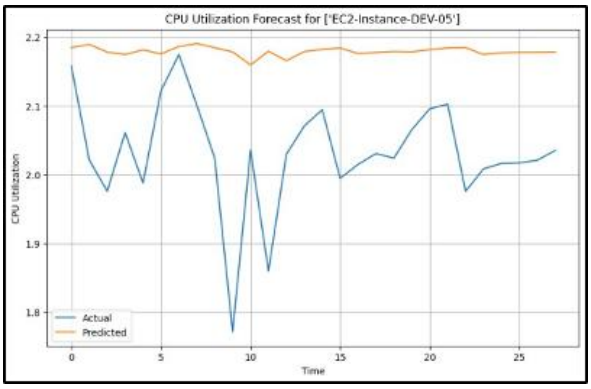
Instance ID	MSE	Training Time (s)
EC2-Instance-DEV-01	0.0321	338.2
EC2-Instance-DEV-02	0.0758	331.8
EC2-Instance-DEV-03	0.1244	348.7
EC2-Instance-DEV-04	0.0048	339.9
EC2-Instance-DEV-05	0.0279	360
EC2-Instance-DEV-06	0.0058	378.6
EC2-Instance-DEV-07	0.3694	434.8
EC2-Instance-DEV-08	0.1005	560.6
EC2-Instance-DEV-09	3.3271	688.9

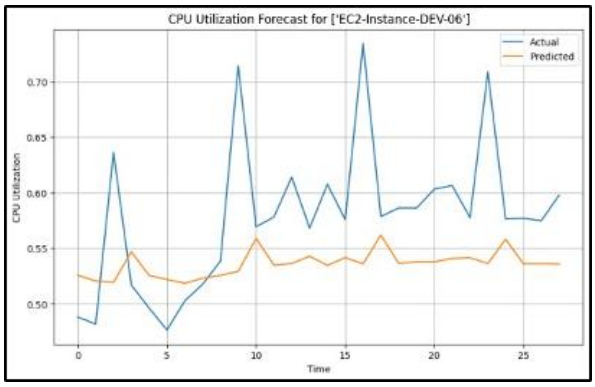

CPU Utilization Forecast for ['EC2-Instance-DEV-01']
Actual
Predicted
CPU Utilization
Time

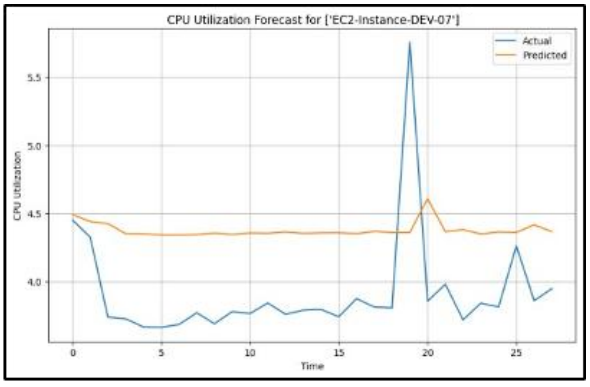

CPU Utilization Forecast for ['EC2-Instance-DEV-02']
Actual
Predicted
CPU Utilization
Time

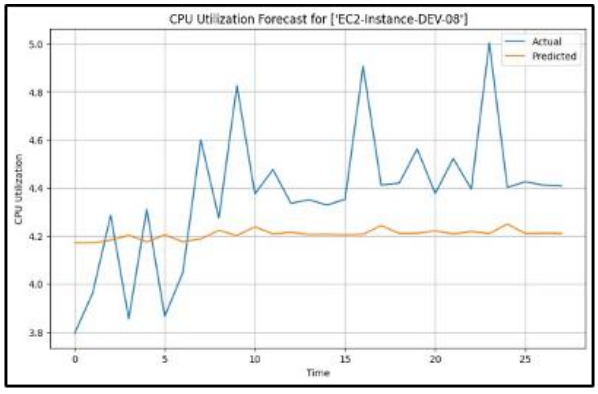

CPU Utilization Forecast for ['EC2-Instance-DEV-03']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-DEV-04']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-DEV-05']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-DEV-06']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-DEV-07']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-DEV-08']
Actual
Predicted
CPU Utilization
Time

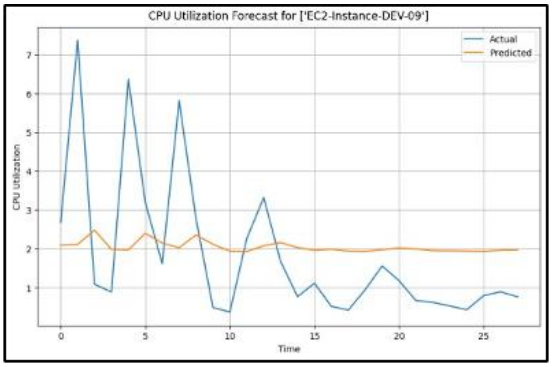
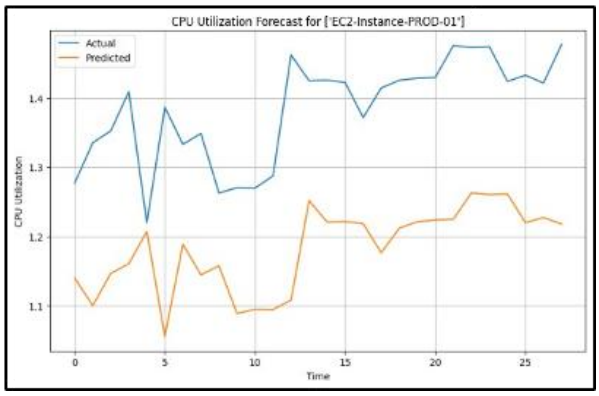


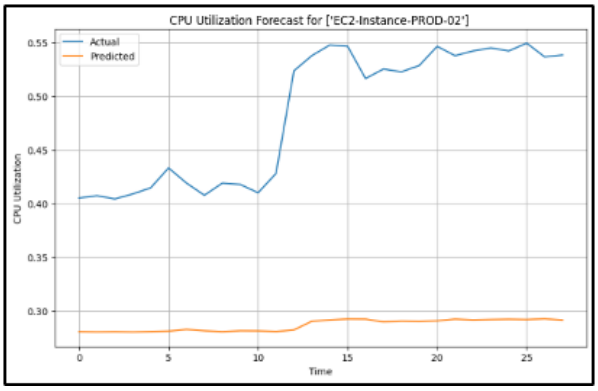
FIGURE 8. Predictive performance of the LSTM models with grid search hyperparameters for the development environment.

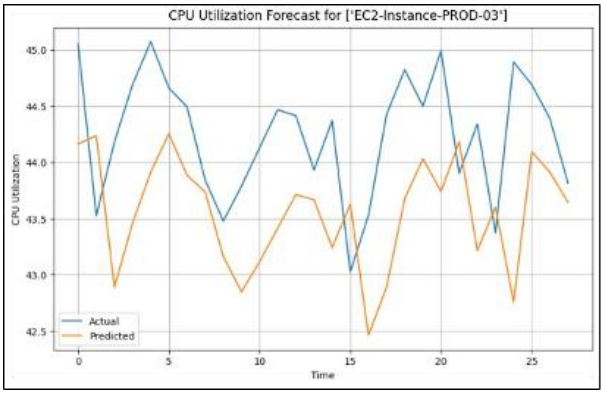
Table 4 demonstrates the results of the grid search optimization in varying improvements for the production environment. For example, EC2-Instance-PROD-05 achieves an MSE of 0.0033 with a training time of 6 minutes and 8.7 seconds. However, EC2-Instance-PROD-03 has a significantly higher MSE of 0.8878, indicating that hyperparameter tuning may not always lead to better performance for all datasets. Figure 10 portrays the predictive performance of the LSTM models with grid search hyperparameters settings for the production environment across various instances.

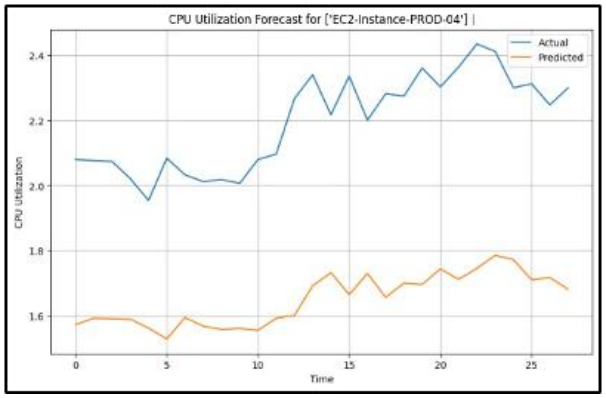
TABLE 4. Results of the LSTM models with grid search hyperparameters for the production environment

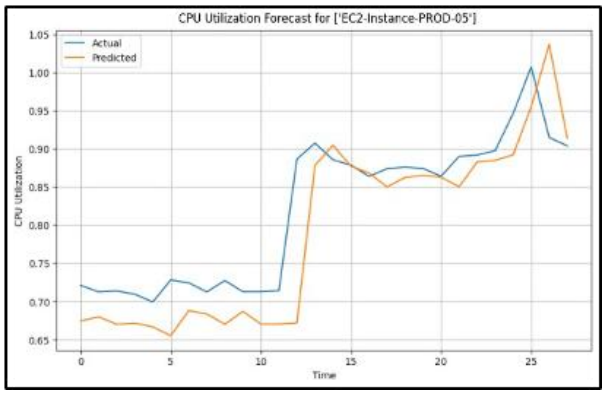
Instance ID	MSE	Training Time (s)
EC2-Instance-PROD-01	0.0443	322.9
EC2-Instance-PROD-02	0.0423	324.9
EC2-Instance-PROD-03	0.8878	332.3
EC2-Instance-PROD-04	0.3053	385.8
EC2-Instance-PROD-05	0.0033	368.7
EC2-Instance-PROD-06	0.2513	355.0
EC2-Instance-PROD-07	0.0029	428.2
EC2-Instance-PROD-08	0.0212	525.0
EC2-Instance-PROD-09	0.1749	793.4

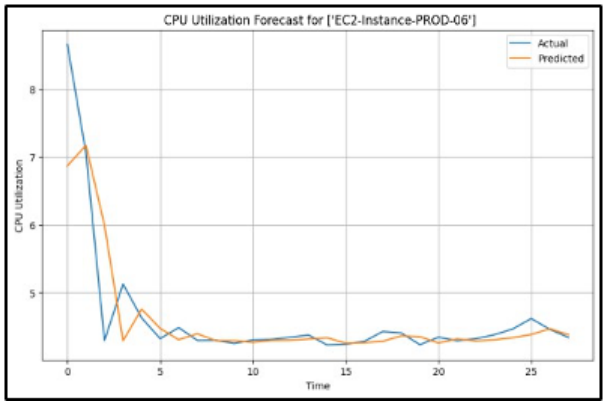

CPU Utilization Forecast for ['EC2-Instance-PROD-01']
Actual
Predicted
CPU Utilization
Time

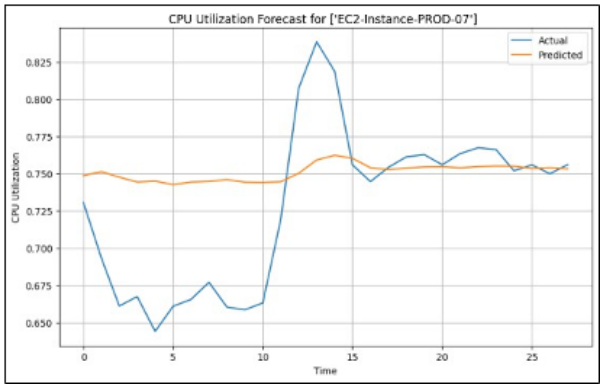

CPU Utilization Forecast for ['EC2-Instance-PROD-02']
Actual
Predicted
CPU Utilization
Time

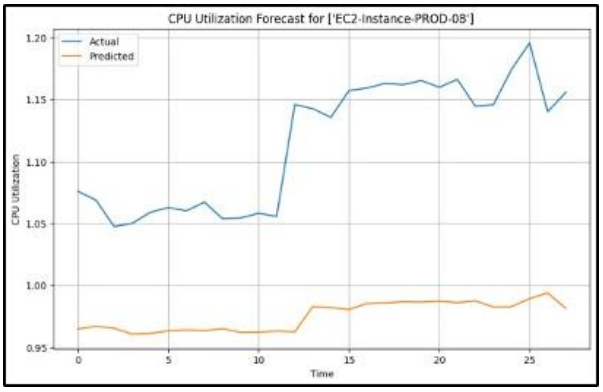

CPU Utilization Forecast for ['EC2-Instance-PROD-03']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-PROD-04']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-PROD-05']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-PROD-06']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-PROD-07']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-PROD-08']
Actual
Predicted
CPU Utilization
Time

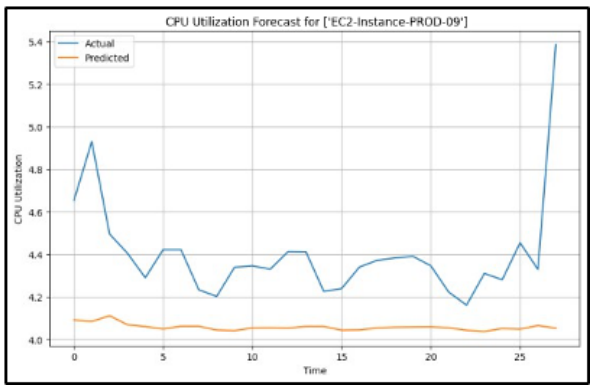


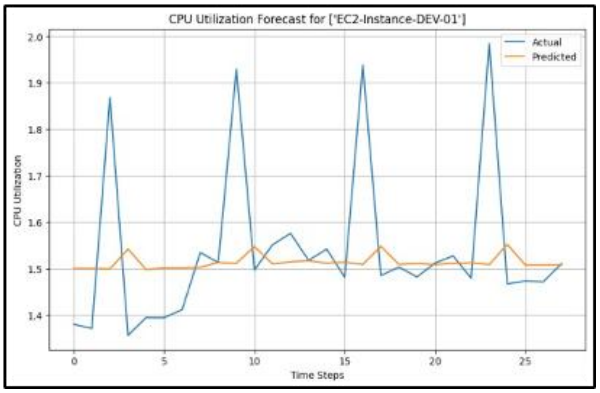
FIGURE 9. Predictive performance of the LSTM models with grid search hyperparameters for the production environment.

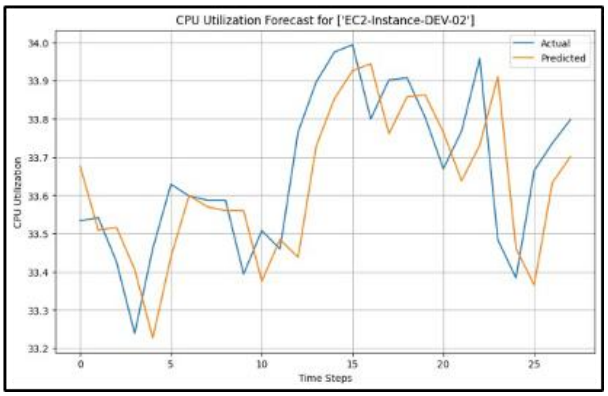
4.1.3 Random Search Hyperparameters

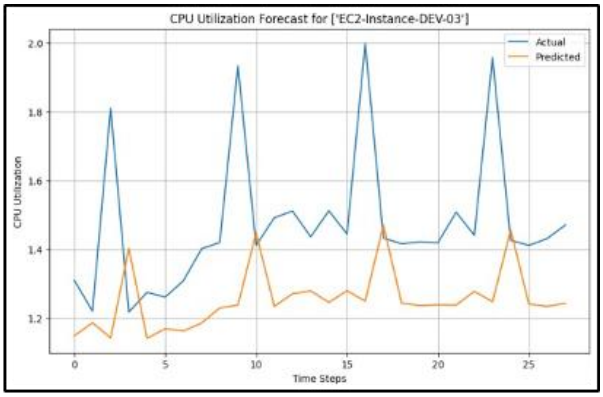
Table 5 provides the results of the random search method for hyperparameter tuning for the development environment. EC2-Instance-DEV-04 achieves an MSE of 0.0051 with a training time of 8.3 seconds, indicating a slight increase in both MSE and training time compared to the default settings. Figure 10 shows the predictive performance of the LSTM models with random search hyperparameters settings for the development environment across various instances.

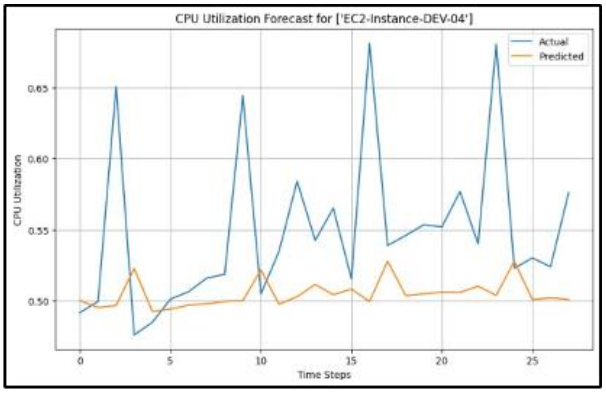
TABLE 5. Results of the LSTM models with random search hyperparameters for the development environment

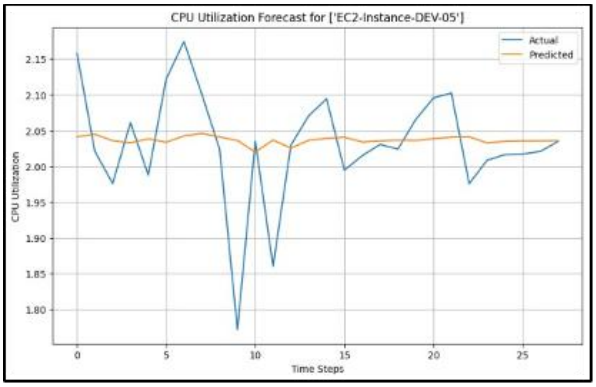
Instance ID	MSE	Training Time (s)
EC2-Instance-DEV-01	0.0302	8.3
EC2-Instance-DEV-02	0.0276	8.2
EC2-Instance-DEV-03	0.0982	7.9
EC2-Instance-DEV-04	0.0051	8.3
EC2-Instance-DEV-05	0.0061	8.3
EC2-Instance-DEV-06	0.0059	8.4
EC2-Instance-DEV-07	0.1709	8.5
EC2-Instance-DEV-08	0.1041	8.0
EC2-Instance-DEV-09	3.3068	8.6

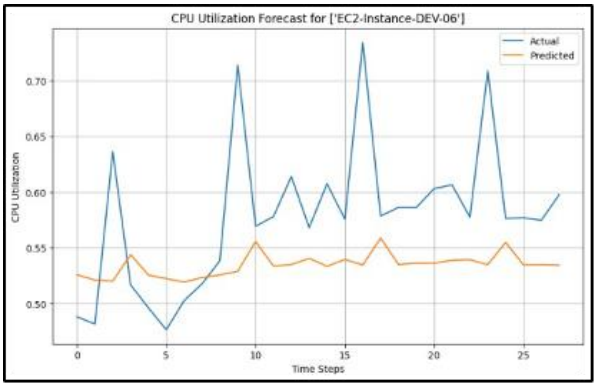

CPU Utilization Forecast for ['EC2-Instance-DEV-01']
Actual
Predicted
CPU Utilization
Time Steps

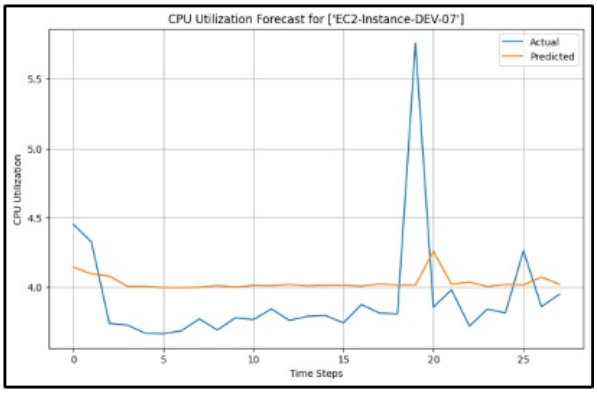

CPU Utilization Forecast for ['EC2-Instance-DEV-02']
Actual
Predicted
CPU Utilization
Time Steps

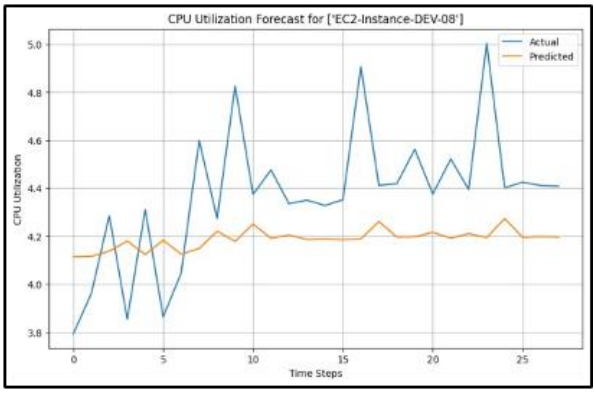

CPU Utilization Forecast for ['EC2-Instance-DEV-03']
Actual
Predicted
CPU Utilization
Time Steps


CPU Utilization Forecast for ['EC2-Instance-DEV-04']
Actual
Predicted
CPU Utilization
Time Steps


CPU Utilization Forecast for ['EC2-Instance-DEV-05']
Actual
Predicted
CPU Utilization
Time Steps


CPU Utilization Forecast for ['EC2-Instance-DEV-06']
Actual
Predicted
CPU Utilization
Time Steps


CPU Utilization Forecast for ['EC2-Instance-DEV-07']
Actual
Predicted
CPU Utilization
Time Steps


CPU Utilization Forecast for ['EC2-Instance-DEV-08']
Actual
Predicted
CPU Utilization
Time Steps

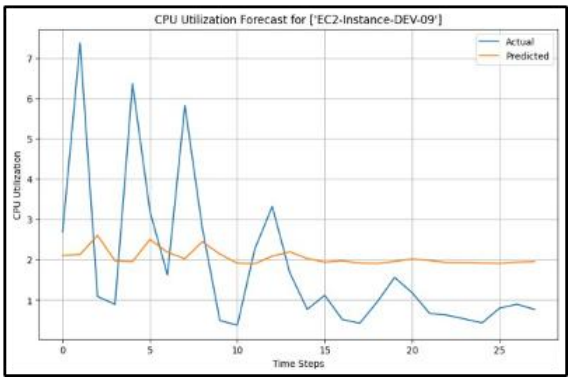
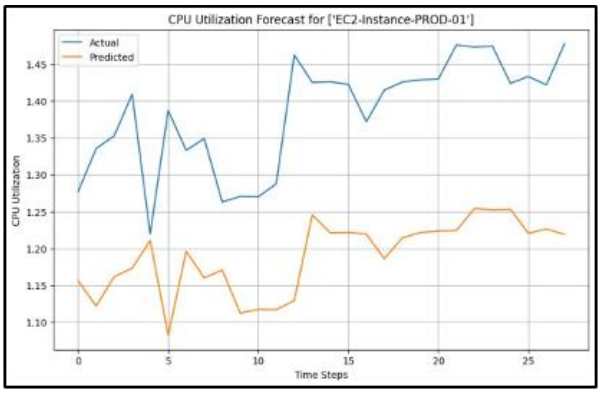


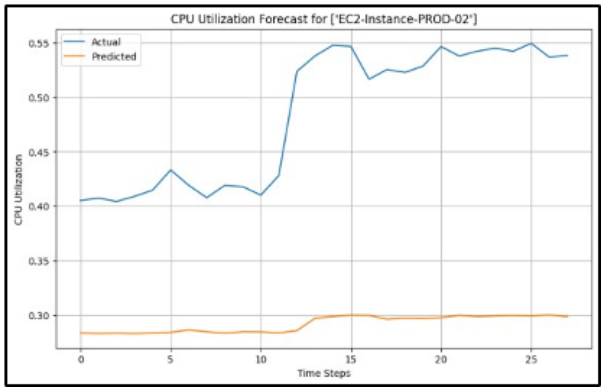
FIGURE 10. Predictive performance of the LSTM models with random search hyperparameters for the development environment.

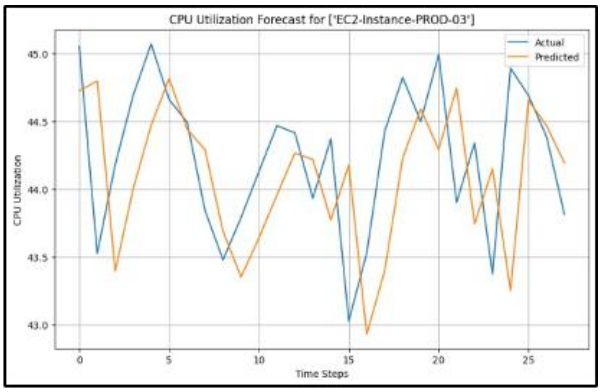
Table 6 shows the results of a random search for the production environment, which shows mixed outcomes. For instance, EC2-Instance-PROD-07 achieves an MSE of 0.0009 with a training time of 8.5 seconds, which is an improvement over the default settings. However, EC2-Instance-PROD-03 still has a high MSE of 0.4541, suggesting that random search may not be as effective in optimizing hyperparameters for this dataset. Figure 11 illustrates the predictive performance of the LSTM models with random search hyperparameters settings for the production environment across various instances.

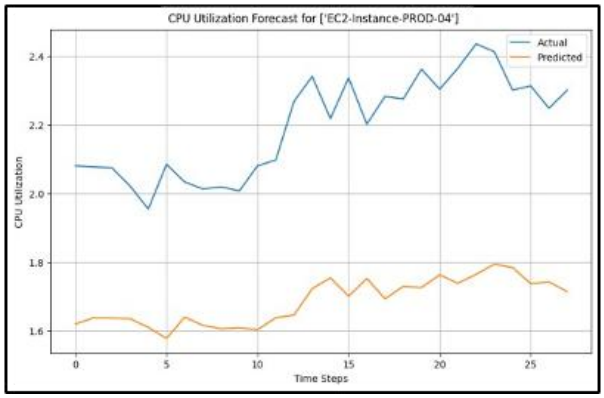
TABLE 6. Results of the LSTM models with random search hyperparameters for the production environment

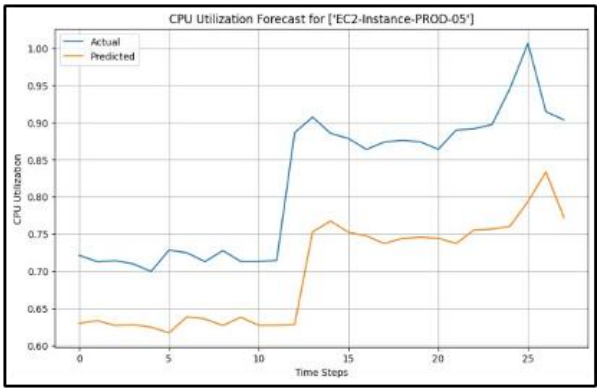
Instance ID	MSE	Training Time (s)
EC2-Instance-PROD-01	0.0413	11.1
EC2-Instance-PROD-02	0.0401	8.2
EC2-Instance-PROD-03	0.4541	8.6
EC2-Instance-PROD-04	0.2699	8.4
EC2-Instance-PROD-05	0.0164	10.0
EC2-Instance-PROD-06	0.2687	8.8
EC2-Instance-PROD-07	0.0012	10.4
EC2-Instance-PROD-08	0.0205	7.9
EC2-Instance-PROD-09	0.0956	8.2

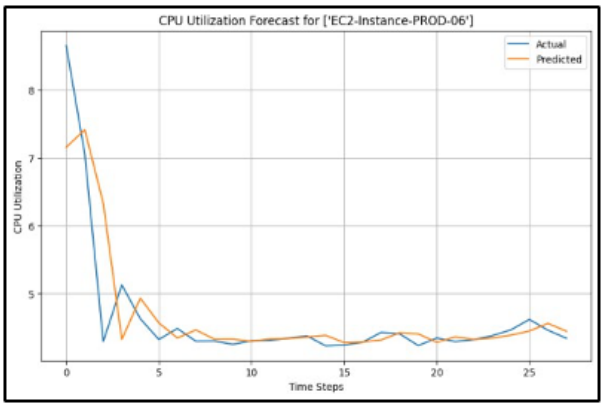

CPU Utilization Forecast for ['EC2-Instance-PROD-01']
Actual
Predicted
CPU Utilization
Time Steps

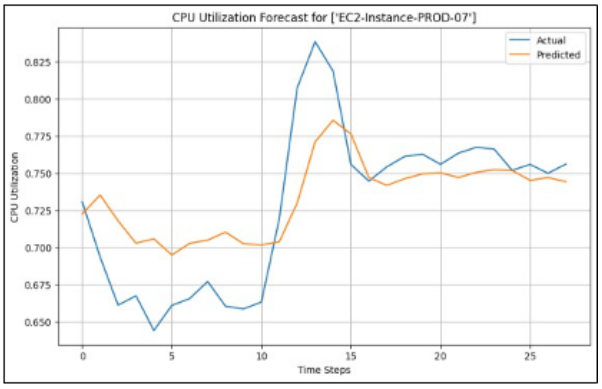

CPU Utilization Forecast for ['EC2-Instance-PROD-02']
Actual
Predicted
CPU Utilization
Time Steps

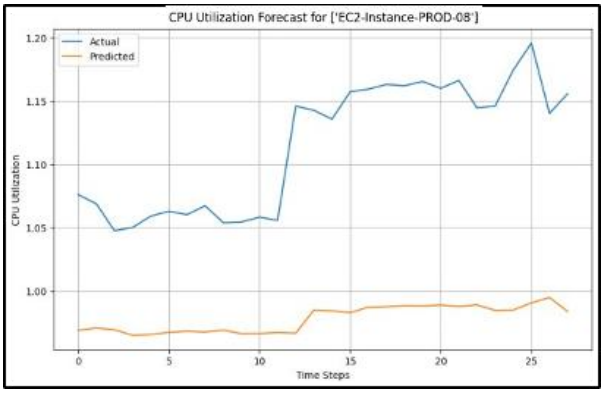

CPU Utilization Forecast for ['EC2-Instance-PROD-03']
Actual
Predicted
CPU Utilization
Time Steps


CPU Utilization Forecast for ['EC2-Instance-PROD-04']
Actual
Predicted
CPU Utilization
Time Steps


CPU Utilization Forecast for ['EC2-Instance-PROD-05']
Actual
Predicted
CPU Utilization
Time Steps


CPU Utilization Forecast for ['EC2-Instance-PROD-06']
Actual
Predicted
CPU Utilization
Time Steps


CPU Utilization Forecast for ['EC2-Instance-PROD-07']
Actual
Predicted
CPU Utilization
Time Steps


CPU Utilization Forecast for ['EC2-Instance-PROD-08']
Actual
Predicted
CPU Utilization
Time Steps

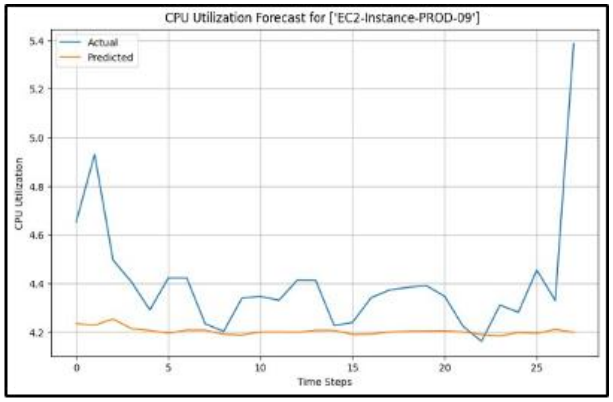


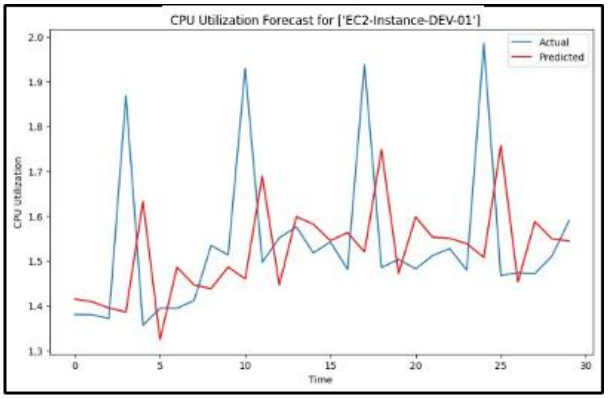
FIGURE 11. Predictive performance of the LSTM models with random search hyperparameters for the production environment.

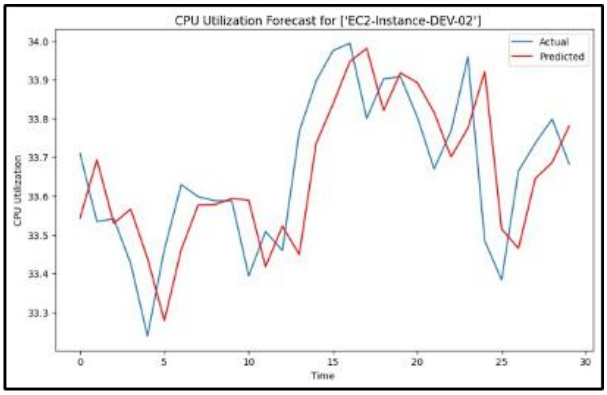
4.2 ARIMA Model Analysis

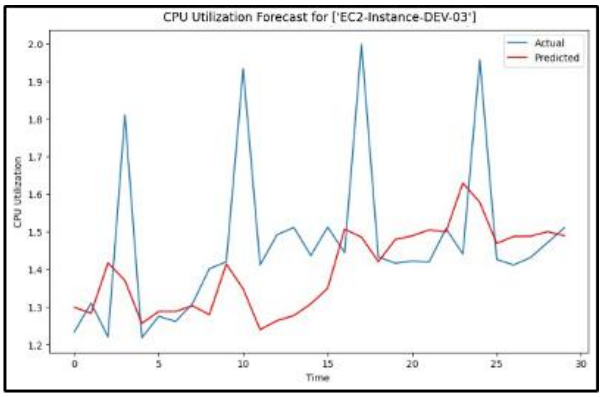
Table 7 provides the results of the ARIMA models for the development environment, which exhibit lower MSE values and shorter fitting times compared to LSTM models. For instance, EC2-Instance-DEV-04 achieves an MSE of 0.0009 with a fitting time of 6.025 seconds, highlighting ARIMA's efficiency in handling this dataset. However, EC2-Instance-DEV-09 has a high MSE of 3.5406, indicating that ARIMA may struggle with certain datasets in the development environment. Figure 12 shows the predictive performance of the ARIMA models for the development environment across various instances.

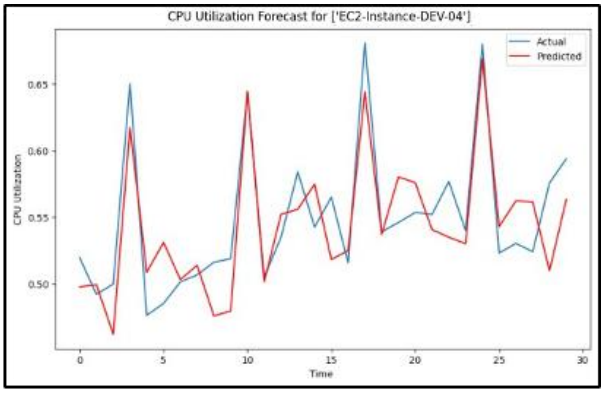
TABLE 7. Results of the ARIMA models for the development environment

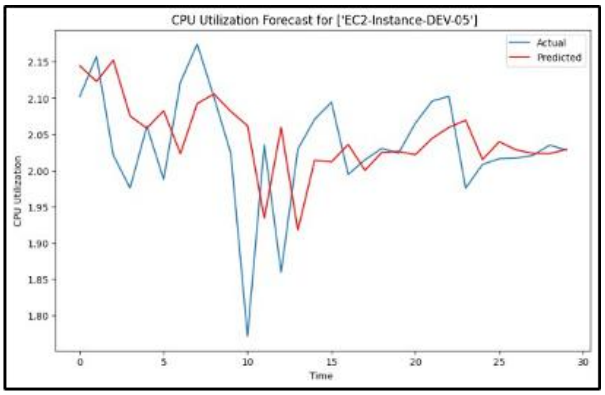
Instance ID	MSE	Fitting Time (s)
EC2-Instance-DEV-01	0.0403	1.925
EC2-Instance-DEV-02	0.0253	0.653
EC2-Instance-DEV-03	0.0417	1.712
EC2-Instance-DEV-04	0.0009	6.025
EC2-Instance-DEV-05	0.0078	1.805
EC2-Instance-DEV-06	0.0037	0.765
EC2-Instance-DEV-07	0.3591	4.647
EC2-Instance-DEV-08	0.1086	1.421
EC2-Instance-DEV-09	3.5406	0.229

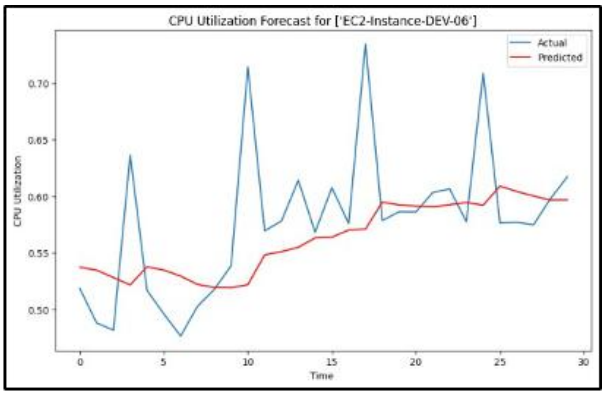

CPU Utilization Forecast for ['EC2-Instance-DEV-01']
Actual
Predicted
CPU Utilization
Time

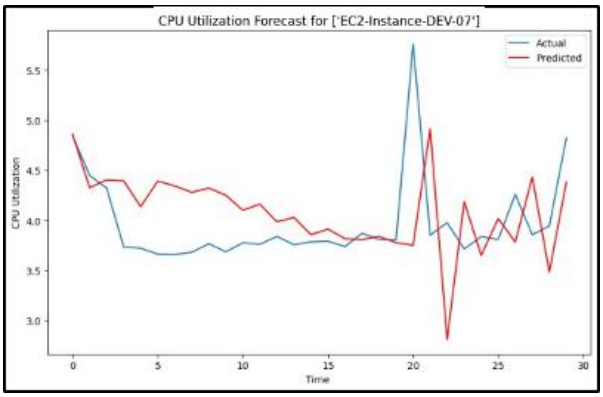

CPU Utilization Forecast for ['EC2-Instance-DEV-02']
Actual
Predicted
CPU Utilization
Time

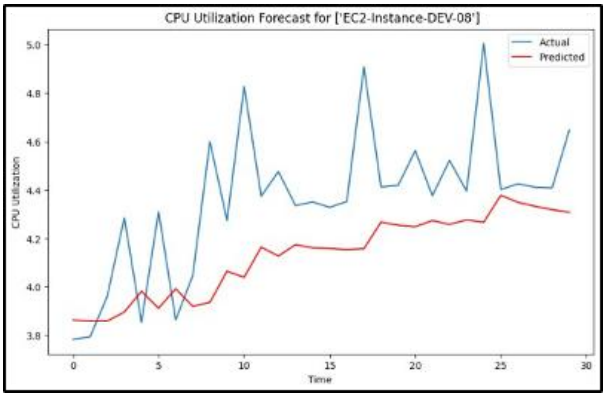

CPU Utilization Forecast for ['EC2-Instance-DEV-03']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-DEV-04']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-DEV-05']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-DEV-06']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-DEV-07']
Actual
Predicted
CPU Utilization
Time


CPU Utilization Forecast for ['EC2-Instance-DEV-08']
Actual
Predicted
CPU Utilization
Time

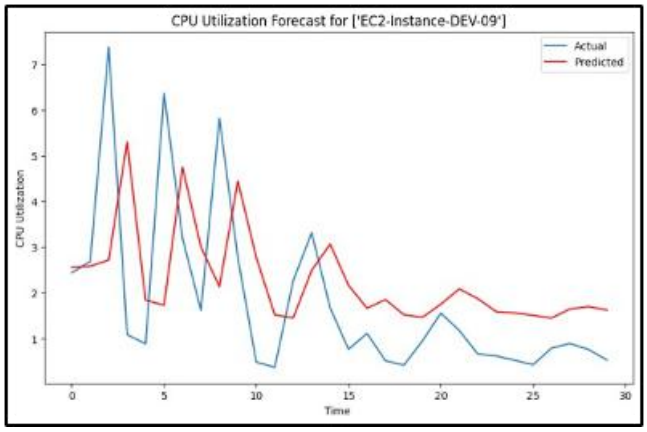
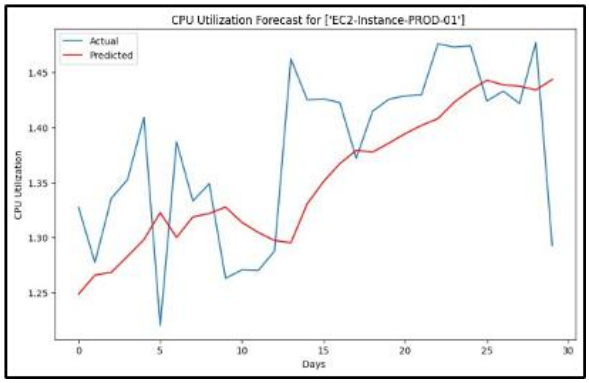


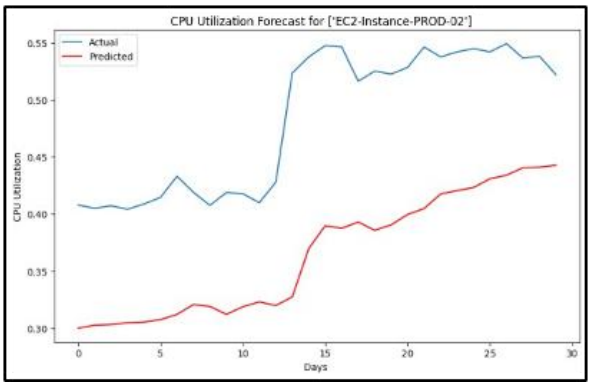
FIGURE 12. Predictive performance of the ARIMA models development environment.

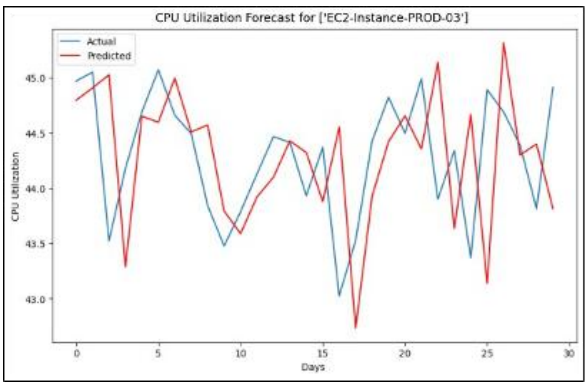
Table 8 presents the results of the ARIMA models for the production environment, maintaining a consistent performance, with low MSE values and relatively fast fitting times. EC2-Instance-PROD-07 achieves an MSE of 0.0016 with a fitting time of 0.409 seconds, demonstrating ARIMA's capability in quickly fitting models to data. However, EC2-Instance-PROD-03 has a higher MSE of 0.5739, indicating some variability in performance. Figure 13 shows the predictive performance of the ARIMA models for the production environment across various instances.

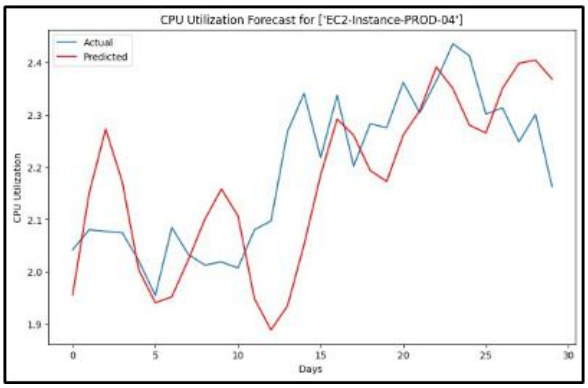
TABLE 8. Results of the ARIMA models for the production environment

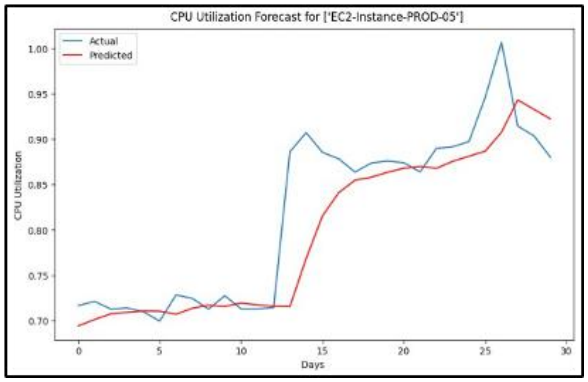
Instance ID	MSE	Fitting Time (s)
EC2-Instance-PROD-01	0.0046	0.512
EC2-Instance-PROD-02	0.0146	2.450
EC2-Instance-PROD-03	0.5739	0.643
EC2-Instance-PROD-04	0.0169	5.664
EC2-Instance-PROD-05	0.0025	0.557
EC2-Instance-PROD-06	0.4428	4.024
EC2-Instance-PROD-07	0.0016	0.409
EC2-Instance-PROD-08	0.0018	1.606
EC2-Instance-PROD-09	0.0865	5.533

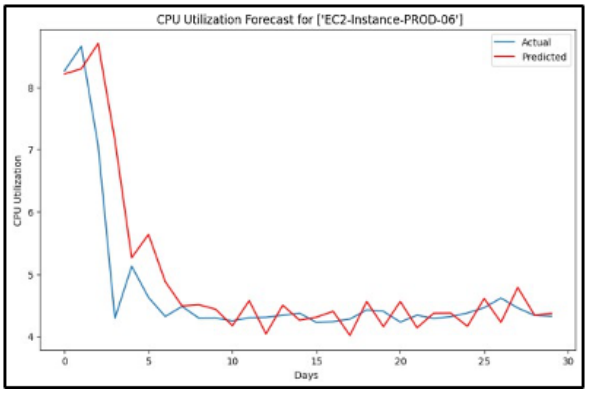

CPU Utilization Forecast for ['EC2-Instance-PROD-01']
Actual
Predicted
CPU Utilization
Days

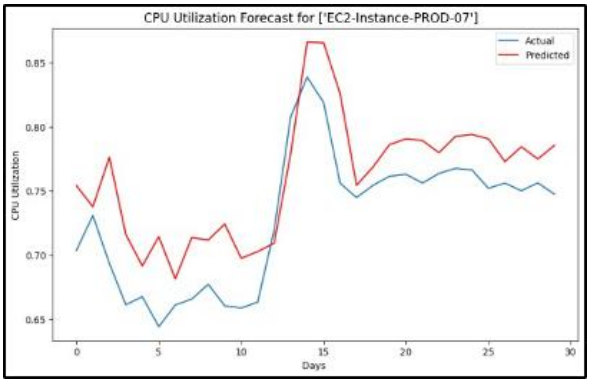

CPU Utilization Forecast for ['EC2-Instance-PROD-02']
Actual
Predicted
CPU Utilization
Days

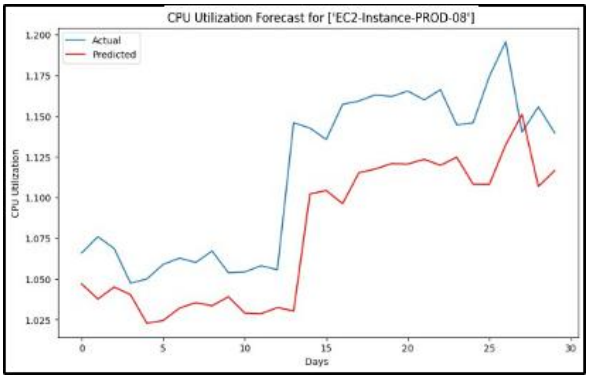

CPU Utilization Forecast for ['EC2-Instance-PROD-03']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-PROD-04']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-PROD-05']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-PROD-06']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-PROD-07']
Actual
Predicted
CPU Utilization
Days


CPU Utilization Forecast for ['EC2-Instance-PROD-08']
Actual
Predicted
CPU Utilization
Days

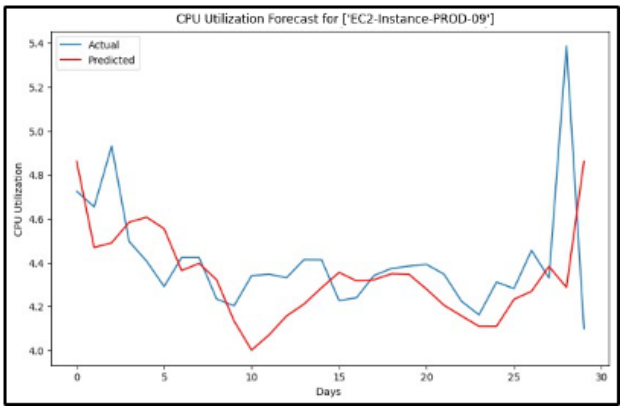


FIGURE 13. Predictive performance of the ARIMA models production environment.

4.3 LSTM Model Performance

The LSTM model showed varied performance based on the hyperparameter tuning methods applied.

Default Hyperparameters: The default configuration provided moderate performance with relatively lower training times. In the production environment, instances such as EC2-Instance-PROD-07 and EC2-Instance-PROD-08 achieved very low MSE values of 0.0008 and 0.0005, respectively, indicating high accuracy.

Grid Search Hyperparameters: This method generally improved the model accuracy at the cost of significantly increased training times. The instance EC2-Instance-PROD-05 in the production environment achieved the lowest MSE of 0.0033, suggesting effective hyperparameter optimization.

Random Search Hyperparameters: This method provided a balance between accuracy and training time. Notably, EC2-Instance-PROD-07 achieved an MSE of 0.0012 in the production environment, with a reasonable training time of 10.4 seconds.

4.4. ARIMA Model Performance

The ARIMA model, known for its statistical robustness, performed differently across instances:

In the development environment, instance EC2-Instance-DEV-04 achieved an exceptionally low MSE of 0.0009, indicating high predictive accuracy with a fitting time of 6.025 seconds.

In the production environment, instance EC2-Instance-PROD-01 had the lowest MSE of 0.0046 with a fitting time of 0.512 seconds, demonstrating both accuracy and efficiency.

4.5. Comparison of LSTM and ARIMA

4.5.1. Model Performance in the Development Environment

In the development environment, ARIMA consistently outperforms all LSTM variants in terms of MSE, particularly for Instance IDs EC2-Instance-DEV-04 (MSE: 0.0009) and EC2-Instance-DEV-06 (MSE: 0.0037). This suggests ARIMA's robustness in handling time series data, which may be attributed to its ability to capture linear patterns effectively.

The LSTM models, while offering competitive performance in some instances, exhibit higher variability in MSE. For example, the LSTM default model achieves a low MSE of 0.0048 for Instance ID EC2-Instance-DEV-04, yet fails to generalize similarly across other instances, such as EC2-Instance-DEV-09, where the MSE soars to 3.2938. The LSTM models with grid search and random search tuning exhibit mixed results. While random search provides an improvement over the default settings in several cases, such for instance ID EC2-Instance-DEV-07 (MSE: 0.1709), it still falls short of ARIMA's performance. Grid search, on the other hand, often results in higher MSE values, indicating potential overfitting or suboptimal parameter selection.

Fitting time is a critical consideration in model selection, especially in time-sensitive or resource-constrained environments. ARIMA models consistently exhibit the shortest fitting times across both environments, typically completing in under 10 seconds, apart from Instance ID EC2-Instance-DEV-04 in the development environment, where the fitting time was slightly higher at 6.025 seconds. This makes ARIMA not only the most accurate but also the most computationally efficient model in this study. The MSE and fitting times for each model in the development environment are summarized in Table 9.

TABLE 9. Combined results for the development environment

Instance ID	LSTM Default MSE	LSTM Grid Search MSE	LSTM Random Search MSE	ARIMA MSE	Fitting time (s)
EC2-Instance-DEV-01	0.0354	0.0321	0.0302	0.0403	5.1
EC2-Instance-DEV-02	0.0285	0.0758	0.0276	0.0253	3.8
EC2-Instance-DEV-03	0.0998	0.1244	0.0982	0.0417	4.6
EC2-Instance-DEV-04	0.0048	0.0048	0.0051	0.0009	6.025
EC2-Instance-DEV-05	0.0086	0.0279	0.0061	0.0078	4.1
EC2-Instance-DEV-06	0.0050	0.0058	0.0059	0.0037	4.4
EC2-Instance-DEV-07	0.4234	0.3694	0.1709	0.3591	9.3
EC2-Instance-DEV-08	0.0884	0.1005	0.1041	0.1086	7.9
EC2-Instance-DEV-09	3.2938	3.3271	3.3068	3.5406	8.6

4.5.2. Model Performance in the Production Environment

The production environment further corroborates ARIMA's superiority, with the model consistently achieving the lowest MSE values across most instances. This indicates that ARIMA outperforms the LSTM models for Instance ID EC2-Instance-PROD-01 (MSE: 0.0046) and EC2-Instance-PROD-08 (MSE: 0.0018), demonstrating its efficacy in real-world forecasting tasks.

The LSTM default model shows moderate success in the production environment, achieving an MSE of 0.0008 for Instance ID EC2-Instance-PROD-07. However, its performance is inconsistent across other instances, and it often lags behind ARIMA. The LSTM grid search and random search models similarly fail to consistently outperform the default LSTM or ARIMA, with grid search yielding the highest MSE values in instances such as EC2-Instance-PROD-03 (MSE: 0.8878).

For fitting time, the LSTM models, particularly those with grid search, demonstrate significantly longer fitting times. For instance, the grid search for Instance ID EC2-Instance-PROD-09 in the production environment took an extensive 793.4 seconds, making it impractical for scenarios requiring quick turnaround times. Even the LSTM models with random search, although faster than grid search, cannot match the fitting efficiency of ARIMA. The MSE and fitting times for each model in the production environment are summarized in Table 10.

TABLE 10. Combined results for the production environment

Instance ID	LSTM Default MSE	LSTM Grid Search MSE	LSTM Random Search MSE	ARIMA MSE	Fitting time (s)
EC2-Instance-PROD-01	0.0278	0.0443	0.0413	0.0046	5.4
EC2-Instance-PROD-02	0.0278	0.0423	0.0401	0.0146	5.2
EC2-Instance-PROD-03	0.4672	0.8878	0.4541	0.5739	10.1
EC2-Instance-PROD-04	0.2613	0.3053	0.2699	0.0169	4.1
EC2-Instance-PROD-05	0.0120	0.0033	0.0164	0.0025	7.9
EC2-Instance-PROD-06	0.2639	0.2513	0.2687	0.4428	8.6
EC2-Instance-PROD-07	0.0008	0.0029	0.0012	0.0016	7.3
EC2-Instance-PROD-08	0.0005	0.0212	0.0205	0.0018	7.1
EC2-Instance-PROD-09	0.1053	0.1749	0.0956	0.0865	793.4

The varied performance between production and development environments suggests that the operating system and the specific configurations of EC2 instances may influence the effectiveness of the forecasting models. Further analysis is required to identify the exact factors

contributing to these differences. The findings of this research are crucial for optimizing resource allocation and predictive maintenance in cloud computing environments. By selecting appropriate forecasting models and tuning methods, organizations can enhance the performance and reliability of their EC2 instances, leading to improved cost efficiency and service delivery.

5. Conclusion

This study presents the analysis of LSTM networks and ARIMA models for forecasting CPU utilization of AWS EC2 instances in both development and production environments. By analyzing a comprehensive dataset from these environments, the research evaluates the effectiveness of these machine learning models in predicting CPU utilization accurately and efficiently. The findings reveal that ARIMA models consistently outperform LSTM networks in several key aspects. ARIMA demonstrates superior predictive accuracy with significantly lower MSE values across various instances. Additionally, ARIMA models achieve shorter fitting times compared to LSTM models, highlighting their efficiency in processing and forecasting time series data. This efficiency is attributed to ARIMA's ability to effectively capture linear patterns within the data, making it a robust choice for forecasting tasks that require both accuracy and computational efficiency. In contrast, while LSTM models have shown potential in time series forecasting, they tend to exhibit higher variability in MSE values. This variability indicates that LSTM models may not consistently generalize well across different instances. The extended fitting times associated with LSTM models, particularly when employing hyperparameter tuning techniques such as grid search and random search, pose practical challenges.

Future research should focus on several key areas to build upon these findings and enhance forecasting capabilities. Firstly, exploring advanced hyperparameter optimization techniques, such as Bayesian optimization, could significantly improve LSTM model performance by more efficiently finding the optimal set of parameters, leading to greater accuracy and reliability. Additionally, investigating ensemble methods that combine multiple forecasting models could provide a more robust solution by leveraging the strengths of each model and addressing their respective weaknesses. This approach may improve overall predictive performance and offer a more comprehensive forecasting strategy.

Funding: This research was financially supported by the University Internal Research Funding (URIF) in collaboration between Universiti Teknologi PETRONAS (UTP) and Bursa Malaysia, with cost centers of 015LB0-093 and 015LB0-095, respectively.

Acknowledgement: The authors extend their deepest gratitude to UTP and Bursa Malaysia for their invaluable support and facilitation during the implementation of this research. Their commitment and assistance were instrumental to the success of this study. The authors deeply

appreciate their contributions and collaborative efforts, which significantly enriched the research process.

Data Availability Statement: The dataset used in this study was obtained from the Malaysia Stock Exchange, Bursa Malaysia Berhad. The data are confidential and were provided exclusively for research purposes. They are not intended for public distribution or sharing, and any use beyond this study is strictly prohibited without the explicit permission of Bursa Malaysia Berhad.

Conflicts of Interest: The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] P.P. Ray, An Introduction to Dew Computing: Definition, Concept and Implications, IEEE Access 6 (2018), 723-737. https://doi.org/10.1109/access.2017.2775042.

[2] S. Goyal, Public Vs Private Vs Hybrid Vs Community - Cloud Computing: A Critical Review, Int. J. Comput. Netw. Inf. Secur. 6 (2014), 20-29. https://doi.org/10.5815/ijcnis.2014.03.03.

[3] A. Suleiman, A. Usman, H. Daud, F.A. Idris, R. Sokkalingam, et al., A Voting Regressor Ensemble Model for Crude Oil Price Prediction, J. Stat. Sci. Comput. Intell. 1 (2025), 61-72. https://doi.org/10.64497/jssci.4.

[4] U. Danjuma Maiwada, R. Yusuf Zakari, A.A. Janisar, Distribution Function-Driven Handover Solutions for 5g Mobile Networks, J. Stat. Sci. Comput. Intell. 1 (2025), 46-60. https://doi.org/10.64497/jssci.1.

[5] A.A. Suleiman, H. Daud, N.S.S. Singh, M. Othman, A.I. Ishaq, et al., A Novel Odd Beta Prime-Logistic Distribution: Desirable Mathematical Properties and Applications to Engineering and Environmental Data, Sustainability 15 (2023), 10239. https://doi.org/10.3390/su151310239.

[6] J. George, Comparing Scalable Serverless Analytics Architecture on Amazon Web Services and Google Cloud, Int. J. Nov. Res. Dev. 9 (2024), a689-a696.

[7] M. Zubair, A.A. Suleiman, A.K. Yousafzai, T. Alazemi, Prediction of Oxide Glass Refractive Index Using a Novel Deep Learning Architecture, Dualnet (U-Net and ANN) with Hybrid Loss Function, Opt. Quantum Electron. 57 (2025), 387. https://doi.org/10.1007/s11082-025-08300-2.

[8] S. Nabi, M. Ibrahim, J.M. Jimenez, DRALBA: Dynamic and Resource Aware Load Balanced Scheduling Approach for Cloud Computing, IEEE Access 9 (2021), 61283-61297. https://doi.org/10.1109/access.2021.3074145.

[9] M. Zubair, H.B. Rais, F. Ullah, A.A. Suleiman, Enhancing Image Reconstruction Fidelity in Low-Dose CT Scans: A Comprehensive MPS-UNet Framework, in: Lecture Notes in Electrical Engineering, Springer Nature Singapore, Singapore, 2025: pp. 59-70. https://doi.org/10.1007/978-981-96-5848-0_6.

[10] A.I. Ishaq, A.U. Usman, H.N. Alqifari, A. Almohaimeed, H. Daud, et al., A New Log-Lomax Distribution, Properties, Stock Price, and Heart Attack Predictions Using Machine Learning Techniques, AIMS Math. 10 (2025), 12761-12807. https://doi.org/10.3934/math.2025575.

[11] L.C. George, Y. Guo, D. Stepanov, V.K. Reddy Peri, et al., Usage Visualisation for the AWS Services, Procedia Comput. Sci. 176 (2020), 3710-3717. https://doi.org/10.1016/j.procs.2020.09.016.

[12] G.A.G. Mireles, M.A. Moraga, F. Garcia, M. Piattini, A Classification Approach of Sustainability Aware Requirements Methods, in: 2017 12th Iberian Conference on Information Systems and Technologies (CISTI), IEEE, 2017, pp. 1-6. https://doi.org/10.23919/cisti.2017.7975813.

[13] A.A. Suleiman, H. Daud, A.G. Usman, S.I. Abba, M. Othman, et al., A New Two-Parameter Half-Logistic Distribution with Numerical Analysis and Applications, J. Stat. Sci. Comput. Intell. 1 (2025), 1-28. https://doi.org/10.64497/jssci.2.

[14] A.G. Usman, S. Mati, H. Daud, A.A. Suleiman, S.I. Abba, et al., Optimized Svr with Nature-Inspired Algorithms for Environmental Modelling of Mycotoxins in Food Virtual-Water Samples, Sci. Rep. 15 (2025), 16569. https://doi.org/10.1038/s41598-025-99908-7.

[15] A.A. Suleiman, H. Daud, A.I. Ishaq, N.S.S. Singh, D.S. Metwally, et al., An Extension of the Gompertz Distribution for Modeling Covid-19 Mortality Dynamics, Int. J. Anal. Appl. 23 (2025), 206. https://doi.org/10.28924/2291-8639-23-2025-206.

[16] A.A. Suleiman, H. Daud, A.I. Ishaq, A.U. Farouk, A.S. Mohammed, et al., A New Statistical Model for Advanced Modeling of Cancer Disease Data, Kuwait J. Sci. 52 (2025), 100429. https://doi.org/10.1016/j.kjs.2025.100429.

[17] U. Panitanarak, A. Ismail Ishaq, A. Adewole Abiodun, H. Daud, A. Abubakar Suleiman, A New Maxwell-Log Logistic Distribution and Its Applications for Mortality Rate Data, J. Niger. Soc. Phys. Sci. (2025), 1976. https://doi.org/10.46481/jnsps.2025.1976.

[18] U. Panitanarak, A.I. Ishaq, N.S.S. Singh, A. Usman, A.U. Usman, et al., Machine Learning Models in Predicting Failure Times Data Using a Novel Version of the Maxwell Model, Eur. J. Stat. 5 (2025), 1. https://doi.org/10.28924/ada/stat.5.1.

[19] A.A. Suleiman, H. Daud, A.I. Ishaq, M. Othman, H.M. Alshanbari, et al., A Novel Extended Kumaraswamy Distribution and Its Application to COVID-19 Data, Eng. Rep. 6 (2024), e12967. https://doi.org/10.1002/eng2.12967.

[20] M. Duggan, K. Mason, J. Duggan, E. Howley, E. Barrett, Predicting Host CPU Utilization in Cloud Computing Using Recurrent Neural Networks, in: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, 2017, pp. 67-72. https://doi.org/10.23919/icitst.2017.8356348.

[21] U. Panitanarak, A.I. Ishaq, A.A. Suleiman, H. Daud, N.S.S. Singh, et al., A New Beta Distribution with Interdisciplinary Data Analysis, AIMS Math. 10 (2025), 8495-8527. https://doi.org/10.3934/math.2025391.

[22] S.S. Panwar, M.M.S. Rauthan, V. Barthwal, A Systematic Review on Effective Energy Utilization Management Strategies in Cloud Data Centers, J. Cloud Comput. 11 (2022), 95. https://doi.org/10.1186/s13677-022-00368-5.

[23] H. Daud, A.S. Mohammed, A.I. Ishaq, B. Abba, Y. Zakari, et al., Modeling and Prediction of Exchange Rates Using Topp-Leone Burr Type X, Machine Learning and Deep Learning Models, Eur. J. Stat. 4 (2024), 11. https://doi.org/10.28924/ada/stat.4.11.

[24] Praveen Borra, Advancing Artificial Intelligence with Aws Machine Learning: A Comprehensive Overview, Int. J. Adv. Res. Sci. Commun. Technol. 4 (2024), 71-78. https://doi.org/10.48175/ijarsct-18810.

[25] M. Balaji, C. Aswani Kumar, G.S.V. Rao, Predictive Cloud Resource Management Framework for Enterprise Workloads, J. King Saud Univ. - Comput. Inf. Sci. 30 (2018), 404-415. https://doi.org/10.1016/j.jksuci.2016.10.005.

[26] L.M. Kaufman, Data Security in the World of Cloud Computing, IEEE Secur. Priv. Mag. 7 (2009), 61-64. https://doi.org/10.1109/msp.2009.87.

[27] A. Ates, K. Suppayah, Disciplined Innovation: A Case Study of the Amazon Working Backwards Approach to Internal Corporate Venturing, Res. Manag. 67 (2024), 23-33. https://doi.org/10.1080/08956308.2024.2326805.

[28] N. Preetham, S. Chhetri, S.R. Kini, G.G. Mishra, S. Kumar, Resource Provisioning in Cloud Using Arima and LSTM Technique, in: 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), IEEE, 2022, pp. 1-7. https://doi.org/10.1109/mysurucon55714.2022.9972689.

[29] T. Nguyen, T. Do, K. Le, S. Go, S. Na, et al., An Lstm-Based Approach for Predicting Resource Utilization in Cloud Computing, in: The 11th International Symposium on Information and Communication Technology, ACM, New York, NY, USA, 2022, pp. 173-179. https://doi.org/10.1145/3568562.3568647.

[30] P. Osypanka, P. Nawrocki, Resource Usage Cost Optimization in Cloud Computing Using Machine Learning, IEEE Trans. Cloud Comput. 10 (2022), 2079-2089. https://doi.org/10.1109/tcc.2020.3015769.

[31] Y. Yu, Y. Jia, M.A. Alshahrani, O.A. Alamri, H. Daud, et al., Adopting a New Sine-Induced Statistical Model and Deep Learning Methods for the Empirical Exploration of the Music and Reliability Data, Alex. Eng. J. 104 (2024), 396-408. https://doi.org/10.1016/j.aej.2024.07.104.

[32] A.A. Suleiman, A.K. Yousafzai, M. Zubair, Comparative Analysis of Machine Learning and Deep Learning Models for Groundwater Potability Classification, 4th Int. Electron. Conf. Appl. Sci. (2023), 249. https://doi.org/10.3390/asec2023-15506.

[33] A.I. Ishaq, A.A. Abiodun, A.A. Suleiman, A. Usman, A.S. Mohammed, et al., Modelling Nigerian Inflation Rates from January 2003 to June 2023 Using Newly Developed Inverse Power Chi-Square Distribution, in: 2023 4th International Conference on Data Analytics for Business and Industry (ICDABI), IEEE, 2023, pp. 644-651. https://doi.org/10.1109/icdabi60145.2023.10629442.

[34] S.F. Salleh, A.A. Suleiman, H. Daud, M. Othman, R. Sokkalingam, et al., Tropically Adapted Passive Building: A Descriptive-Analytical Approach Using Multiple Linear Regression and Probability Models to Predict Indoor Temperature, Sustainability 15 (2023), 13647. https://doi.org/10.3390/su151813647.

[35] A.A. Suleiman, H. Daud, M. Othman, A. Husin, A.I. Ishaq, et al., Forecasting the Southeast Asian Currencies Against the British Pound Sterling Using Probability Distributions, Data Sci. Insights 1 (2023), 31-51. https://doi.org/10.63017/jdsi.v1i1.5.

[36] A.A. Suleiman, M. Othman, H. Daud, M.L. Abdullah, E.A. Kadir, et al., Forecasting the Volatility of Real Residential Property Prices in Malaysia: A Comparison of Garch Models, Real Estate Manag. Valuat. 31 (2023), 20-31. https://doi.org/10.2478/remav-2023-0018.

[37] M. Othman, R. Indawati, A.A. Suleiman, M.B. Qomaruddin, R. Sokkalingam, Model Forecasting Development for Dengue Fever Incidence in Surabaya City Using Time Series Analysis, Processes 10 (2022), 2454. https://doi.org/10.3390/pr10112454.

[38] A.A. Suleiman, A. Suleiman, U.A. Abdullahi, S.A. Suleiman, Estimation of the Case Fatality Rate of COVID-19 Epidemiological Data in Nigeria Using Statistical Regression Analysis, Biosaf. Health 3 (2021), 4-7. https://doi.org/10.1016/j.bsheal.2020.09.003.

[39] D. Janardhanan, E. Barrett, CPU Workload Forecasting of Machines in Data Centers Using LSTM Recurrent Neural Networks and ARIMA Models, in: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, 2017. https://doi.org/10.23919/icitst.2017.8356346.

[40] X. Le, H.V. Ho, G. Lee, S. Jung, Application of Long Short-Term Memory (lstm) Neural Network for Flood Forecasting, Water 11 (2019), 1387. https://doi.org/10.3390/w11071387.

[41] X. Le, H.V. Ho, G. Lee, S. Jung, Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting, Water 11 (2019), 1387. https://doi.org/10.3390/w11071387.

[42] S. Phawandee, N. Sangurai, N. Putpuek, N. Cooharojananone, A Comparative Analysis of Deep Learning Approaches for Fire Hotspots Prediction in Thailand, in: 2024 23rd International Symposium on Communications and Information Technologies (ISCIT), IEEE, 2024, pp. 134-139. https://doi.org/10.1109/ISCIT63075.2024.10793656.

[43] M. Ayitey Junior, P. Appiahene, O. Appiah, Forex Market Forecasting with Two-Layer Stacked Long Short-Term Memory Neural Network (LSTM) and Correlation Analysis, J. Electr. Syst. Inf. Technol. 9 (2022), 14. https://doi.org/10.1186/s43067-022-00054-1.

[44] H. Alibrahim, S.A. Ludwig, Hyperparameter Optimization: Comparing Genetic Algorithm Against Grid Search and Bayesian Optimization, in: 2021 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2021, pp. 1551-1559. https://doi.org/10.1109/cec45853.2021.9504761.

[45] A. Nugroho, H. Suhartanto, Hyper-parameter Tuning Based on Random Search for Densenet Optimization, in: 2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), IEEE, 2020, pp. 96-99. https://doi.org/10.1109/icitacee50144.2020.9239164.