

Time Series Network Utilization KPI Forecasting Using Advanced AI/ML Models

Niraj Gadhe¹, Vinay Saini², Kirti Bhardwaj³, Moulik Jain⁴, and Shubhi Sharma⁵

^{1,2,3,4,5}*Independent Researcher*

Abstract

The rapid proliferation of data-intensive applications, cloud infrastructure, and IoT ecosystems has made proactive resource provisioning critical for maintaining optimal network performance. However, network administrators face a constant battle against capacity constraints, where traditional reactive approaches fail to accurately anticipate traffic fluctuations. This inability to foresee demand leads to costly over-provisioning, unexpected downtime, and degraded quality of service—directly impacting operational budgets and business continuity. To achieve efficient capacity planning, accurate forecasting of bandwidth utilization is essential. This study addresses the challenge by evaluating a diverse spectrum of models—including seasonal decomposition, Prophet, Random Forest, XGBoost, Support Vector Regression, and advanced deep learning architectures like bidirectional and Convolutional LSTMs—using a common interface dataset benchmarked across MAPE, NRMSE, and R^2 metrics. Ultimately, this research delivers actionable insights into the trade-offs between model accuracy and computational efficiency, empowering engineers, operators, and business owners to select the optimal forecasting model for their specific infrastructure needs.

Keywords: time series forecasting, network utilization, bandwidth prediction, LSTM, XGBoost, Prophet, machine learning

1 Introduction

With the rapid increase in data-intensive applications, cloud infrastructure, and IoT ecosystems [1], network administrators are challenged to ensure optimal performance, minimizing congestion, and proactively provisioning resources [1]. Accurate forecasting of network resources is therefore essential for efficient capacity planning and maintaining high quality of service [2].

While predictive techniques can be applied to various network metrics such as traffic, latency, jitter, and packet loss, this paper specifically focuses on forecasting bandwidth utilization—a key metric for efficient network operations [3]. To achieve this, we explore a range of predictive models including Seasonal Decomposition and Prophet (Additive Time Series Model) [4], tree-based models like Random Forest and XGBoost, as well as supervised regression models like Support Vector Regression (SVR) [5]. Additionally, this research incorporates advanced time series analysis techniques and architecture, utilizing stacked and bidirectional Long Short-Term Memory (LSTM) networks, neural network-based models such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), to effectively capture complex temporal dynamics in network performance [1].

Each algorithm is applied to a common interface utilization dataset and evaluated using key performance metrics such as Mean Absolute Percentage Error (MAPE), Normalized Root Mean Squared Error (NRMSE), and R^2 to assess model accuracy and computational efficiency [6]. Ultimately, this research aims to provide actionable insights into the strengths and limitations of different forecasting methods, helping stakeholders such as Technical Engineers, Business Owners and Network Operators select the most effective models for their specific bandwidth forecasting needs.

2 Challenges

Driven by the accelerated evolution of technology, we are leveraging Artificial Intelligence (AI) within this data-centric environment to transform information into a catalyst for growth, efficiency, and reliability.

However, increasing complexity and scale of AI applications have surfaced several challenges, prompting us to conduct this study. One of the key issues we encountered was the inability to accurately predict bandwidth usage, which led to inefficient capacity planning, underutilized resources in some regions, and congestion or service degradation in other regions. These challenges were compounded by the inconsistent data collected from multiple sources. Missing data due to faulty connections and other underlying problems, noisy data, and outliers further caused another barrier [7].

Beyond the data itself, the high computational need for advanced AI models, particularly in the training phase, required substantial resources—processing power, storage, and expert involvement [8]. As our interactions and data volume grew, we realized existing infrastructure lacked the stability to handle real-time traffic [9].

Moreover, many of the models lacked transparency. Real-time processing emerged as another crucial need—we needed models which can adjust according to real traffic and provide accurate prediction over time [10]. Lastly, maintaining this model over time proved challenging due to concept drift. Monitoring and retraining became essential to ensure sustained accuracy [11]. Coupled with the rising cost of running and maintaining this model, it became clear that deeper study was needed. These practical and technical obstacles led us to pursue a more systematic and intelligent approach to time series forecasting using AI [11].

3 Methodology and Approach

This section outlines the fundamental components and methodologies integral to developing robust AI/ML models for forecasting network utilization of KPIs [12]. We explore suitable algorithms, model architectures, and problem-framing approaches, including illustrative charts and tables, that lay the groundwork for effective prediction [13].

3.1 Candidate Algorithms and Models

The selection of an appropriate algorithm or model architecture is crucial for accurate time series forecasting [12]. The choice often depends on the specific characteristics of the network utilization data, the desired forecast horizon, the complexity of patterns, and the availability of computational resources [1].

Some of the key categories considered for this research paper are given below:

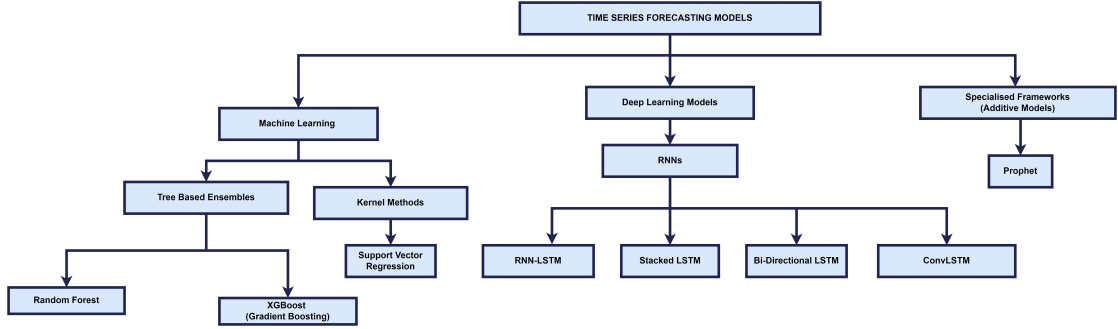


Figure 1: Categorization of Candidate Time Series Forecasting Models/Algorithms.

3.2 Problem Formulation and Analytical Framework

Effectively framing the forecasting problem and utilizing appropriate analytical tools and visualizations are foundational to understanding the data and developing accurate models.

Table 1: Defining the Forecasting Task: Indicators and Parameters.

Parameter	Description
Identifying Key Performance Indicators (KPIs) for Forecasting	The initial step is to explicitly identify the specific network metrics that require prediction. The choice of KPIs will depend on operational objectives, such as capacity planning, anomaly detection, or resource optimization [2]. For our current analysis, we are using Network Bandwidth Utilization [3]. <i>Examples:</i> Average/Peak Link Bandwidth Utilization (%), Data Throughput (e.g., Gbps, Mbps), Number of Active Connections/Sessions, Network Latency (ms), Packet Loss or Error Rates (%)
Establishing the Forecast Horizon	This parameter defines the future time period for which predictions are needed. The forecast horizon can range from very short-term (e.g., minutes) to long-term (e.g., days or weeks) [2]. The length of the horizon significantly impacts model choice, complexity, and the potential for accumulating forecast error [14]. <i>Horizons:</i> Short-term: Next 15 minutes, 1 hour; Medium-term: Next 24 hours, 7 days; Long-term: Next 30 days, 1 quarter
Specifying Data Granularity	Data granularity refers to the time resolution of both the input data used for model training and the desired forecast output [15]. Finer granularity can capture more detailed patterns but may also introduce more noise, while coarser granularity might smooth over crucial short-term fluctuations [15]. <i>Granularities:</i> High-frequency: Per-second, 1-minute intervals; Standard: 5-minute, 15-minute, hourly intervals; Low-frequency: daily or weekly intervals

3.3 System and Design Requirements

For this study, the system and design requirements specify the use of Jupyter notebooks as the primary computing environment, with CSV files serving as the main format for data storage and exchange. The implementation relies on Python 3.8 and a diverse set of libraries, including NumPy, Pandas, Matplotlib, TensorFlow, Keras, Scikit-learn, XGBoost, Prophet, and Statsmodels, to facilitate data preprocessing, visualization, and model training. Additionally, the system configuration requires 16 GB of RAM and 128 GB of disk space.

4 Experiments and Analysis

This section outlines the fundamental components and methodologies integral to developing robust AI/ML models for forecasting network utilization of KPIs. We explore suitable algorithms, model architectures, and problem-framing approaches, including illustrative charts and tables, that lay the groundwork for effective prediction.

4.1 Data Collection and Pre-processing

4.1.1 Data Collection

Data is collected from various sources further normalized into a Key Performance Indicator (KPI). For evaluating our results, we have used data from 4 different types of interfaces with the data points being sampled uniformly at a fixed 15-minute resolution.

4.1.2 Dataset Analysis and Preview

In the dataset, represented in Table 2, each row captures a snapshot at a specific timestamp, recorded under the `datetime` field, paired with a Key Performance Indicator (KPI) that describes what is being measured, such as utilization rates. The `index` field identifies the unique network interface, and the `value` field contains the actual numeric measurement for the specified time, interface, and KPI. Using this structure, the dataset enables detailed network performance analyses over time, including investigations of bandwidth usage, capacity planning, and traffic flow by device or interface.

Additionally, Table 3 outlines general functions of network interface types used in the creation of the dataset.

Table 2: Dataset Preview.

Date & Time	KPI	INDEX	VALUE
19-06-2019 06:55	Interface_Utilization	Interface_01	21582.61
19-06-2019 07:10	Interface_Utilization	Interface_01	21443.13
19-06-2019 07:25	Interface_Utilization	Interface_01	19211.76
19-06-2019 07:40	Interface_Utilization	Interface_02	19785.06
19-06-2019 07:55	Interface_Utilization	Interface_02	20128.95
19-06-2019 08:10	Interface_Utilization	Interface_03	20774.73
19-06-2019 08:25	Interface_Utilization	Interface_03	18825.38
19-06-2019 08:40	Interface_Utilization	Interface_04	18162.39
19-06-2019 08:55	Interface_Utilization	Interface_04	16488.59
19-06-2019 09:10	Interface_Utilization	Interface_04	16912.57

Table 3: Interface Type and Function.

Interface Label	Interface Type	Function
Interface_01	Edge Peering Interface	Connects internal networks to external networks or peers, facilitating efficient routing and secure data exchange.
Interface_02	Caching Interface	Enhances performance by storing frequently accessed data locally, reducing latency and bandwidth usage.
Interface_03	Core-to-Edge Interface	Connects core networks to edge networks, enabling high-speed data transfer and processing across network segments.
Interface_04	Mobile Network Interface	Supports mobile data transmission, ensuring efficient access to network services for mobile devices.

4.1.3 Data Cleaning and Visualization

The preprocessing steps include the removal of null values from data. The DateTime index in our dataset is set as the primary index, and data is resampled to an interval of 15 minutes to standardize the time series [7]. Plotting of interface values is performed to visualize the data, enabling a better understanding of underlying patterns and potential anomalies [16].

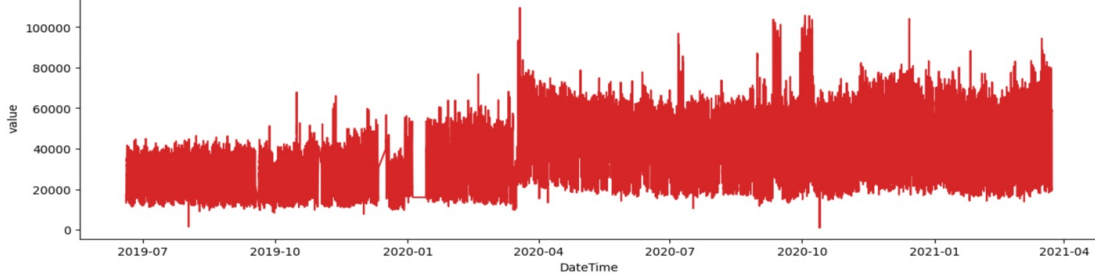


Figure 2: Interface utilization for interface 1.

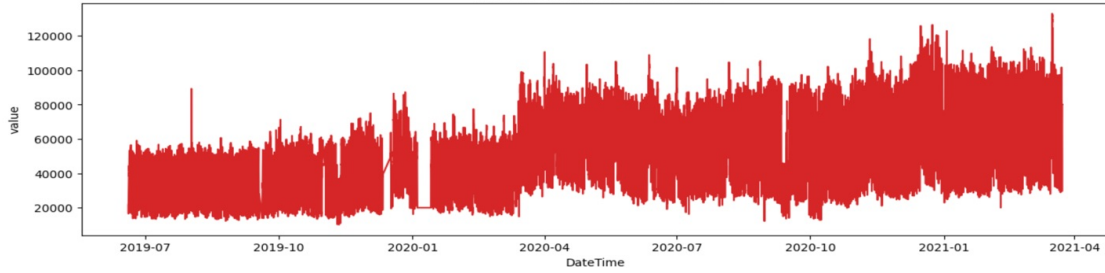


Figure 3: Interface utilization for interface 2.

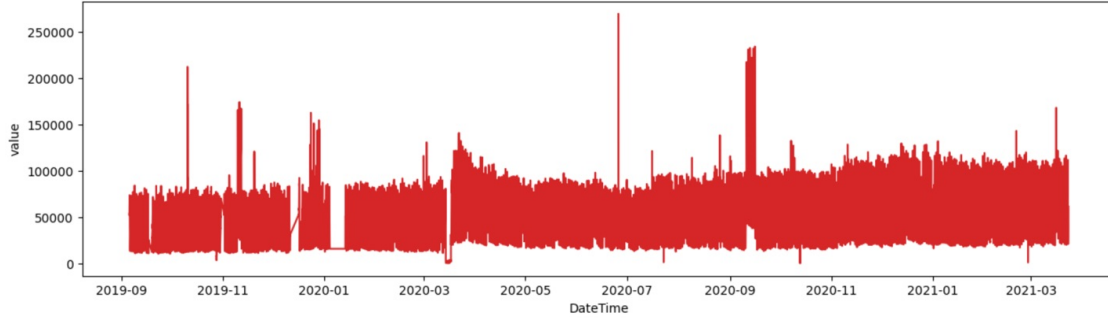


Figure 4: Interface utilization for interface 3.

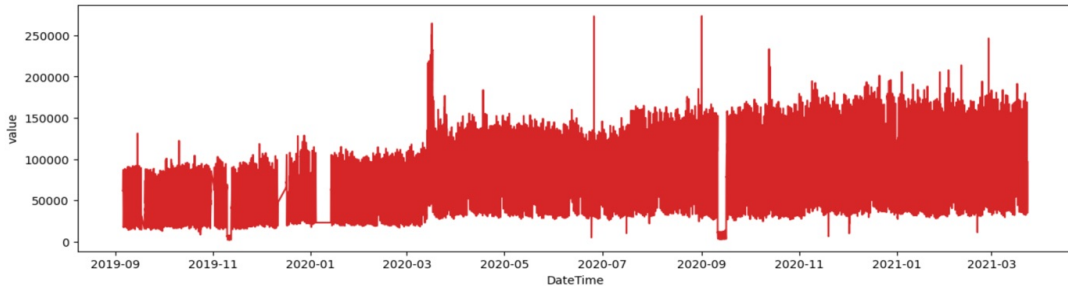


Figure 5: Interface utilization for interface 4.

4.1.4 Seasonality, Trend and Periodicity

In network bandwidth analysis, understanding seasonality, trend, and periodicity is crucial for accurate forecasting and trend identification. Seasonality refers to recurring patterns or fluctuations that occur at regular intervals, often influenced by external factors such as time of day, day of week, or seasonal variations [17]. Trend represents the long-term direction or trajectory of network bandwidth usage, indicating overall growth or decline over time [18].

Periodicity refers to repetitive cycles or oscillations within the data, which may arise from periodic events or systemic patterns [19].

By analyzing and modeling these components, we can uncover underlying patterns, anticipate future trends, and adapt network management strategies accordingly. Seasonality, trend, and periodicity analysis serve as fundamental pillars in network bandwidth forecasting, enabling proactive resource allocation, capacity planning, and performance optimization [2].

The dataset used in this experiment is a univariate time series data, focusing on a single KPI over time [20]. Our analysis indicates that the data exhibits no significant seasonal patterns across interfaces. However, we observe a discernible trend within the data, characterized by a slight increase in bandwidth utilization over time. By acknowledging these characteristics, we tailor our forecasting models.

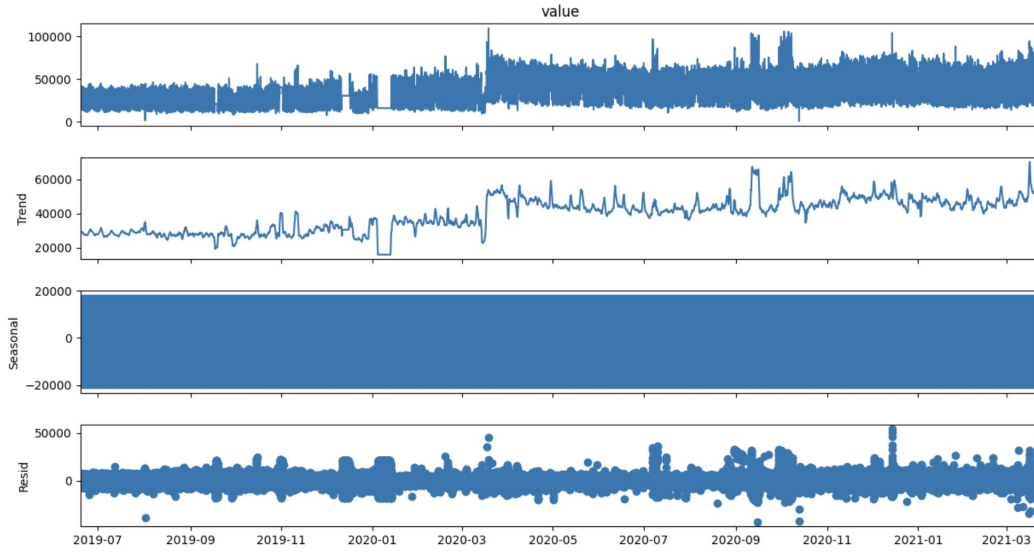


Figure 6: Seasonal decomposition for Interface 1.

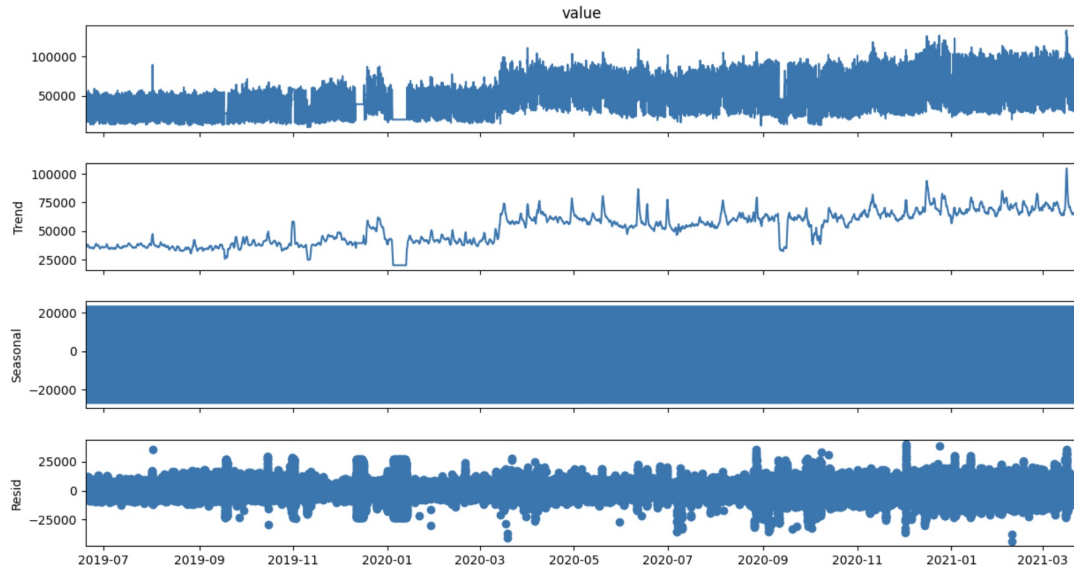


Figure 7: Seasonal decomposition for Interface 2.

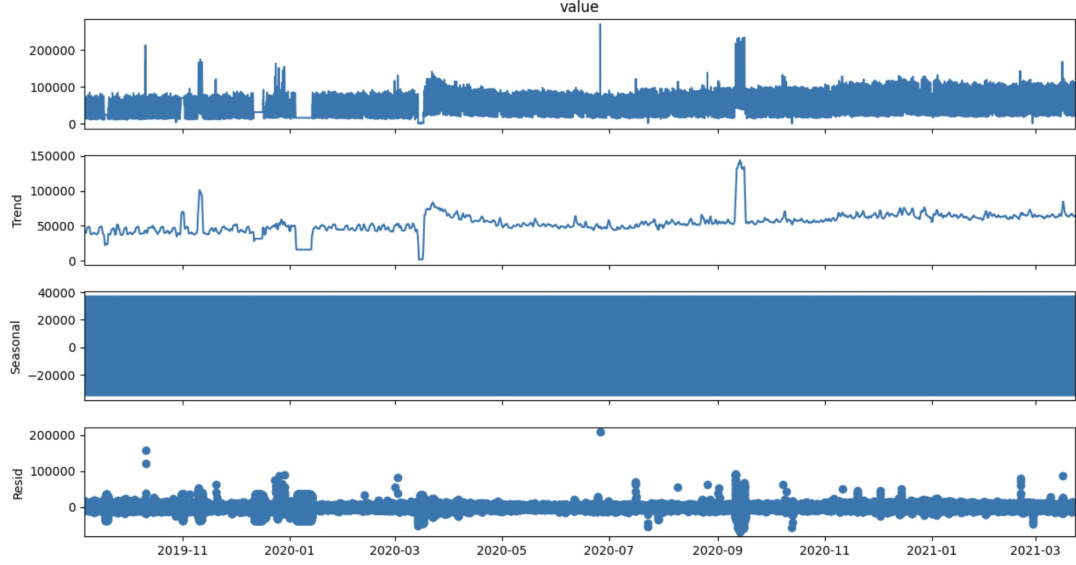


Figure 8: Seasonal decomposition for Interface 3.

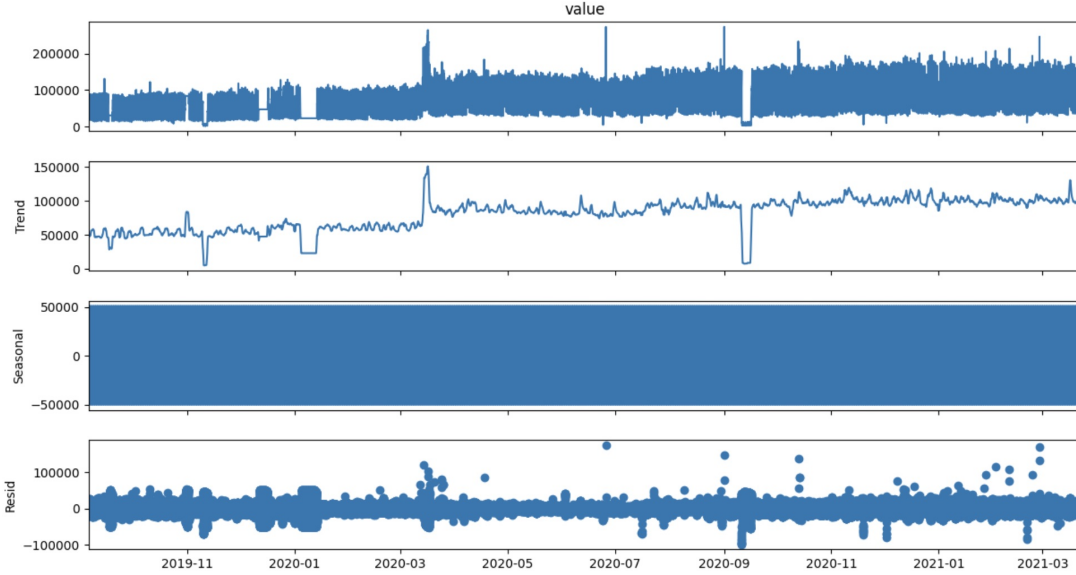


Figure 9: Seasonal decomposition for Interface 4.

4.2 Forecasting Algorithms with Hyperparameters

In network bandwidth forecasting, we employed a diverse array of forecasting algorithms, each tailored to capture specific nuance and pattern in data. In the pursuit of optimizing the performance and predictive accuracy of our forecasting models, careful tuning of hyperparameters plays a crucial role [13]. In this section, we outline the key hyperparameters associated with each forecasting algorithm employed in our analysis.

4.2.1 Random Forest

The `RandomForestRegressor` model was trained without any hyperparameters being explicitly specified or tuned. By default, the model utilizes a set of default hyperparameter values provided by the `RandomForestRegressor` class in scikit-learn.

Table 4: Hyperparameters for Random Forest Model.

Hyperparameter	Description
<code>n_estimators</code>	100 (Number of trees in the forest)
<code>criterion</code>	<code>mse</code> (Mean Squared Error, used to measure the quality of a split)
<code>max_depth</code>	<code>None</code> (Nodes are expanded until all leaves are pure or contain fewer samples)
<code>min_samples_split</code>	2 (Minimum number of samples required to split an internal node)
<code>min_samples_leaf</code>	1 (Minimum number of samples required to be at a leaf node)
<code>bootstrap</code>	<code>True</code> (Bootstrap samples are used when building trees)
<code>random_state</code>	42 (Ensures reproducibility of results)

4.2.2 XGBoost

The XGBoost model was trained using default hyperparameters without any explicit tuning. XGBoost employs a set of default hyperparameter values, including parameters related to tree boosting, regularization, and optimization algorithms.

4.2.3 FBProphet

The Prophet model was trained using default hyperparameters without any explicit tuning.

4.2.4 SVR

The parameters used include a regularization parameter (C) set to 5, a radial basis function (RBF) kernel with a gamma value of 0.5, and an epsilon value of 0.05. Additionally, the model's degree was set to 5, indicating a polynomial kernel function. Other hyperparameters such as cache size, coefficient 0, maximum iterations, and tolerance were set to their default values provided by the Support Vector Regression implementation in scikit-learn.

4.2.5 Convolutional LSTM

This model comprises a stack of layers, including a `ConvLSTM2D` layer with 64 filters and a (1, 1) kernel size, followed by `Flatten` and two `Dense` layers with 32 neurons each. The model is trained using the Adam optimizer and mean squared error loss. This architecture effectively captures spatiotemporal dependencies in the data, facilitating accurate forecasting of network bandwidth utilization.

Model: "sequential"

Layer (type)	Output Shape	Param #
<code>conv_lstm2d (ConvLSTM2D)</code>	<code>(None, 1, 1, 64)</code>	19200
<code>flatten (Flatten)</code>	<code>(None, 64)</code>	0
<code>dense (Dense)</code>	<code>(None, 32)</code>	2080
<code>dense_1 (Dense)</code>	<code>(None, 1)</code>	33
Total params: 21313 (83.25 KB)		
Trainable params: 21313 (83.25 KB)		
Non-trainable params: 0 (0.00 Byte)		

Figure 10: Convolutional LSTM model structure.

4.2.6 Vanilla/RNN LSTM

The model consists of an LSTM layer with 64 units, followed by a Dense layer with 64 units and ReLU activation, and a final Dense output layer with linear activation. The model is

compiled with the Adam optimizer and mean absolute error loss function. This architecture effectively captures temporal dependencies in the data, facilitating accurate forecasting of network bandwidth utilization.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 64)	19200
dense_2 (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 1)	65
Total params: 23425 (91.50 KB)		
Trainable params: 23425 (91.50 KB)		
Non-trainable params: 0 (0.00 Byte)		

Figure 11: RNN LSTM model structure.

4.2.7 Stacked LSTM

This architecture includes two LSTM layers with 64 units each and ReLU activation, configured to return sequences from the first layer. Subsequently, a Dense layer with 64 units and hyperbolic tangent activation is added, followed by a final Dense output layer with linear activation. The model is compiled using the Adam optimizer and mean squared error loss function.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, None, 64)	19200
lstm_2 (LSTM)	(None, 64)	33024
dense_4 (Dense)	(None, 64)	4160
dense_5 (Dense)	(None, 1)	65
Total params: 56449 (220.50 KB)		
Trainable params: 56449 (220.50 KB)		
Non-trainable params: 0 (0.00 Byte)		

Figure 12: Stacked LSTM model structure.

4.2.8 Bi-Directional LSTM

This architecture utilizes a Bidirectional wrapper around an LSTM layer with 64 units and ReLU activation. The model processes input sequences bidirectionally, capturing dependencies from both past and future contexts. A Dense layer with 64 units and hyperbolic tangent activation is added, followed by a final Dense output layer with hyperbolic tangent activation. The model is compiled using the Adam optimizer and mean squared error loss function.

Model: "sequential_3"

Layer (type)	Output Shape	Param #
bidirectional (Bidirectional)	(None, 128)	38400
dense_7 (Dense)	(None, 1)	129
Total params: 38529 (150.50 KB)		
Trainable params: 38529 (150.50 KB)		
Non-trainable params: 0 (0.00 Byte)		

Figure 13: Bi-directional LSTM model structure.

4.3 Model Training

4.3.1 Training

We used data from four distinct interfaces, employing varying sample sizes ranging from 1000 to 50000 data points from each interface [3]. This diverse dataset selection allowed us to comprehensively capture various aspects of network traffic behavior and performance across multiple interfaces [3]. To ensure robust model evaluation and generalization, we employed a consistent training and testing split ratio of 70:30 across all dataset sizes [21].

For each algorithm, the input data was structured in a time-series format, consisting of consecutive observations of network bandwidth utilization recorded at discrete time intervals [12]. The dataset was organized into sequences of timestamps (t_1, t_2, t_3, t_4) representing historical observations, with the subsequent timestamp (t_5) reserved for model training [22]. Notably, the sequence size for input varied for each algorithm, accommodating the specific requirements of the modeling techniques employed [23]. During the training of all LSTM models and Support Vector Regression (SVR), a data normalization process was implemented to scale the input features within the range of $(0, 1)$ [24]. This normalization was essential to ensure compatibility with the activation functions utilized by these models, namely hyperbolic tangent (tanh) for LSTM models and rectified linear unit (ReLU).

4.3.2 Performance Metrics

To assess model accuracy and enable comparative analysis, we used the following key metrics. These helped determine which models performed best across both accuracy and computational efficiency.

Table 5: Key Performance Metrics for Model Evaluation.

Metric	Description
Mean Absolute Percentage Error (MAPE)	Calculates the average percentage error between predicted and actual values, offering insight into the relative accuracy of predictions across different scales [25].
Normalized Root Mean Squared Error (NRMSE)	Normalizes RMSE by the range of the target variable, offering a scale-independent accuracy metric [26].
Training Time	The time required to train a model on the dataset, reflecting its computational cost and scalability [27].
Prediction Time	The time taken to generate forecasts on unseen data, indicating the model's efficiency during inference [9].
R^2 Metric	Indicates the proportion of variance in the dependent variable that is predictable from the independent variables, assessing the goodness of fit for a model [28].

To evaluate model performance, we considered both absolute values and relative comparisons across these metrics [10]. Lower values of MAPE and NRMSE indicate better predictive accuracy, while shorter training and prediction times reflect higher computational efficiency [29]. For MAPE and NRMSE, the following thresholds were used to interpret model accuracy [30]:

- $< 10\%$: High accuracy
- $10\text{--}20\%$: Moderate accuracy
- $20\text{--}30\%$: Acceptable
- $> 30\%$: Unacceptable

5 Evaluation

Each model was evaluated using the following metrics to identify the most accurate and efficient model for bandwidth utilization forecasting. The results of each are given below.

5.1 Random Forest

A reliable traditional model with consistent results.

- MAPE was consistently $< 5\%$, confirming high accuracy.
- NRMSE was $\leq 22\%$ for most interfaces.
- Training time was constant, maxing at 44 seconds per run.
- R^2 remained above 95% across interfaces, with Interface 3 highest and Interface 4 most variable.
- Prediction time remained very low at small sample sizes (~ 0.01 – 0.06 s) and increased with larger sizes, reaching up to ~ 0.57 s.

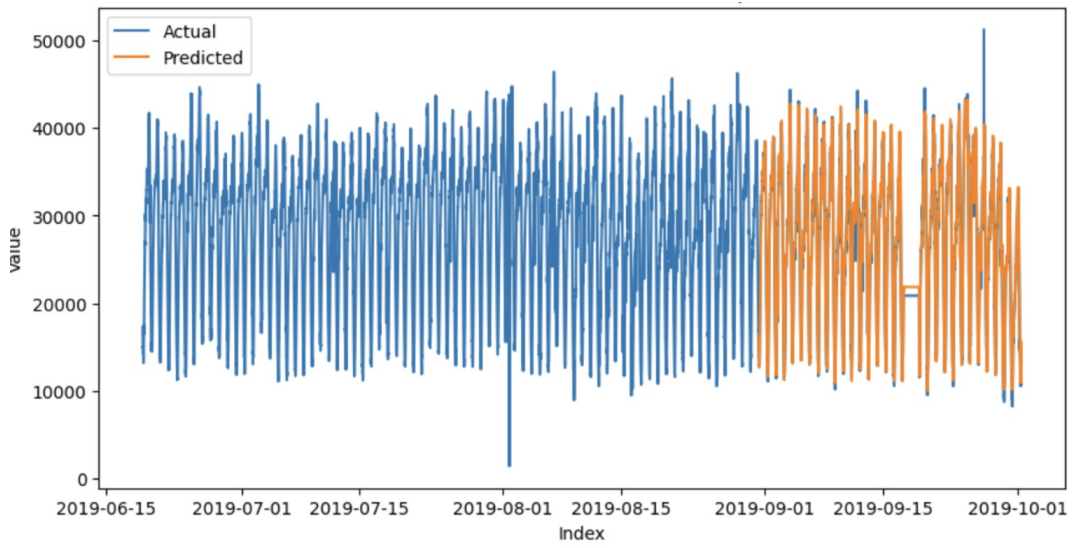


Figure 14: Random Forest: actual vs. predicted values.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	2.4342	2.8732	2.7311	3.9896
2000	2.106	2.525	2.8867	3.0703
5000	4.2217	2.6941	2.8421	3.4708
10000	2.8003	3.1881	2.9658	2.9534
20000	3.5728	3.2524	31.5338	4.016
40000	3.0614	3.0302	4.7463	3.3463
50000	4.2468	4.1538	3.866	3.8661

Figure 15: Comparison of MAPE on all interfaces for Random Forest.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	11.3796	12.1654	8.3242	22.54
2000	9.1264	8.9829	9.3481	9.8307
5000	19.0375	13.7995	9.3633	12.1886
10000	12.2961	14.3286	13.948	11.1653
20000	11.3098	14.9694	11.6829	47.4762
40000	15.2222	15.2808	21.4283	11.2532
50000	22.7328	19.6721	17.3666	12.0292

Figure 16: Comparison of NRMSE on all interfaces for Random Forest.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.7128	0.5401	0.8113	0.5269
2000	1.082	1.0491	0.9026	1.2506
5000	2.8894	2.8763	2.7092	2.9742
10000	7.4645	7.2875	7.0056	6.6349
20000	15.5213	15.1165	13.8059	13.2277
40000	33.6664	33.3249	33.0608	33.1057
50000	44.5335	44.5714	43.9858	44.106

Figure 17: Comparison of training time on all interfaces for Random Forest.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.0124	0.0121	0.0122	0.0115
2000	0.0177	0.035	0.0185	0.0348
5000	0.0353	0.0351	0.0601	0.0354
10000	0.0689	0.079	0.1112	0.0648
20000	0.1233	0.1293	0.1464	0.2058
40000	0.4444	0.2838	0.5717	0.3382
50000	0.4232	0.4314	0.4497	0.3709

Figure 18: Comparison of prediction time on all interfaces for Random Forest.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	98.6697	98.4912	99.3252	95.3883
2000	99.1821	99.2218	99.1009	98.9814
5000	96.4506	98.1435	99.1114	98.5022
10000	98.5532	98.0678	98.0169	98.8259
20000	98.8317	97.9678	98.7308	84.6151
40000	97.6621	97.6555	95.86	98.744
50000	95.2068	96.1878	97.2995	98.577

Figure 19: Comparison of R^2 metric on all interfaces for Random Forest.

5.2 XGBoost

High-performing and extremely fast, ideal for real-time scenarios.

- MAPE was $< 5\%$, and NRMSE ranged from 5–27%.
- Training time was < 1 second, highlighting scalability and speed.
- NRMSE ranged from $\sim 8.78\%$ to $\sim 45.73\%$.
- Prediction time for XGBoost stayed minimal at small sample sizes (~ 0.003 – 0.009 s) and grew moderately with larger datasets.

- R^2 remained mostly high ($\sim 94\text{--}99\%$), with Interface 3 performing strongest overall.

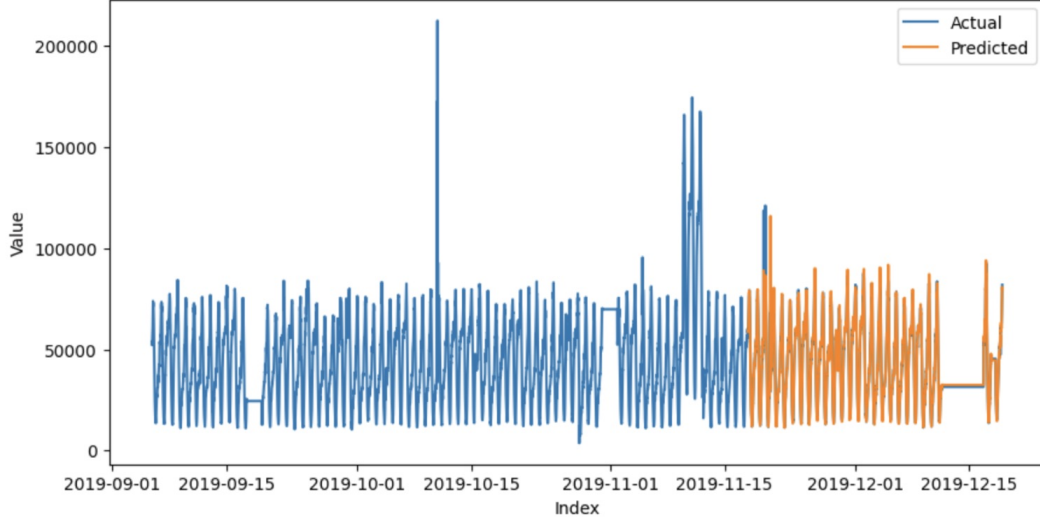


Figure 20: XGBoost: actual vs. predicted values for different sample sizes.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	2.5889	3.2328	2.9154	4.1343
2000	2.2502	2.7411	3.1956	3.4075
5000	4.4082	2.9218	2.9918	3.583
10000	2.9035	3.348	2.8379	2.9843
20000	3.5807	3.875	31.4093	4.2137
40000	3.1535	3.2183	5.2273	3.4384
50000	4.4514	4.4286	4.1951	3.9569

Figure 21: Comparison of MAPE on all interfaces for XGBoost.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	11.6241	13.6251	8.7788	23.4855
2000	9.9415	9.7527	10.7228	11.1798
5000	19.1444	14.3711	9.5349	12.7302
10000	12.4776	14.8553	14.8451	11.6665
20000	12.8253	18.0664	14.8908	45.731
40000	15.9517	16.4033	27.1731	12.1611
50000	26.3834	20.4462	22.7142	13.4125

Figure 22: Comparison of NRMSE on all interfaces for XGBoost.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.2086	0.1763	0.1807	0.2066
2000	0.234	0.2092	0.2182	0.1939
5000	0.2663	0.2609	0.2241	0.2477
10000	0.9247	0.2753	0.2699	1.2108
20000	0.3651	0.3502	1.4919	0.3076
40000	0.4314	0.3942	0.4677	0.4038
50000	0.4584	0.4287	0.5207	0.5119

Figure 23: Comparison of training time on all interfaces for XGBoost.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.009	0.0032	0.0032	0.0029
2000	0.007	0.0037	0.0036	0.0037
5000	0.0058	0.0052	0.0052	0.0068
10000	0.0262	0.0088	0.0154	0.009
20000	0.0142	0.0239	0.015	0.0144
40000	0.0285	0.0257	0.0259	0.027
50000	0.0392	0.0314	0.0665	0.0316

Figure 24: Comparison of prediction time on all interfaces for XGBoost.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	98.6192	98.1052	99.2295	95.076
2000	99.0277	99.0719	98.8939	98.727
5000	96.4266	97.9796	99.0973	98.4015
10000	98.4893	97.8925	97.8043	98.7031
20000	98.3912	97.1942	97.8719	85.8814
40000	97.4501	97.3323	94.0272	98.5347
50000	93.8019	95.9076	95.5375	98.158

Figure 25: Comparison of R^2 on all interfaces for XGBoost.

5.3 Prophet

Underperformed in current configuration; needs tuning.

- MAPE and NRMSE were mostly $> 20\%$, indicating poor accuracy.
- Training time was fast for small data, but averaged ~ 35 seconds for larger datasets.
- NRMSE showed high variability (~ 0.03 – 199.88%), with Interface 3 performing best overall at most sample sizes, while Interfaces 1 and 2 had occasional large spikes.
- R^2 varied widely across interfaces, ranging from -331.68% to 94.24% , with Interface 4 generally performing best and Interface 2 showing the most negative values at larger sample sizes.
- Prediction time for Prophet increased with sample size, ranging from ~ 0.16 s to ~ 9.62 s.
- Requires hyperparameter tuning for better results.

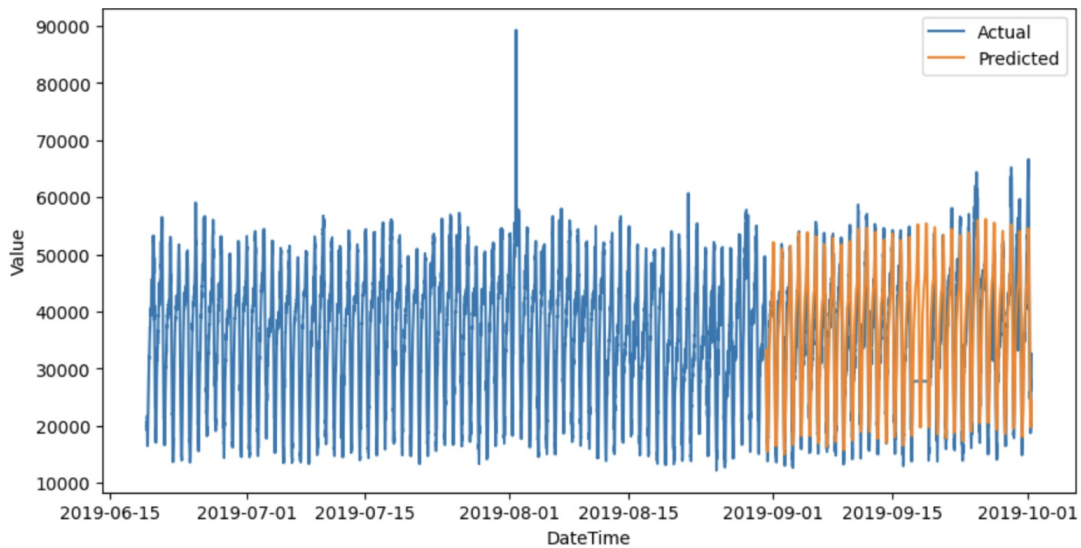


Figure 26: Prophet: actual vs. predicted values.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	20.3012	25.5122	24.7248	20.5061
2000	7.7387	11.6455	25.8399	18.4576
5000	7.5479	5.8326	12.1347	11.2497
10000	22.2417	12.5092	30.8582	95.3961
20000	50.8125	37.521	251.0185	28.9489
40000	66.398	64.9599	22.1496	53.7096
50000	17.5045	19.87	24.7747	75.5801

Figure 27: Comparison of MAPE on all interfaces for Prophet.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	72.0941	84.6794	68.4611	64.9492
2000	25.9751	40.9661	84.7762	62.3764
5000	33.94	32.121	32.578	33.0692
10000	76.7291	51.833	73.1051	199.8788
20000	153.0633	155.0868	111.9898	99.2447
40000	190.3153	188.1987	125.1154	107.1842
50000	111.3791	112.5975	93.8087	106.7775

Figure 28: Comparison of NRMSE on all interfaces for Prophet.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.7357	0.2017	0.1819	0.2212
2000	0.3783	0.4188	0.3135	0.3428
5000	2.46	1.0163	1.0794	2.2311
10000	1.593	2.557	3.1054	2.5531
20000	9.18	6.4934	8.7994	6.7021
40000	26.8605	34.2737	27.562	29.9818
50000	40.3809	35.8552	33.0797	28.7971

Figure 29: Comparison of training time on all interfaces for Prophet.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.19	0.1735	0.1642	0.1907
2000	0.2999	0.34	0.3096	0.3244
5000	0.7587	0.8063	0.8691	1.2756
10000	1.9564	2.7026	2.261	1.5874
20000	4.6179	3.0165	3.2378	4.4901
40000	7.056	6.4751	8.1097	6.5961
50000	9.6177	9.4134	7.9194	9.1333

Figure 30: Comparison of prediction time on all interfaces for Prophet.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	39.8362	17.3567	61.7978	67.0702
2000	93.1857	84.5054	46.921	70.7355
5000	88.29	89.19	58.29	89.66
10000	84.19	91.72	47.1909	-371.5676
20000	49.56	-29.6464	38.56	56.07
40000	-251.1788	-331.6807	82.65	94.24
50000	79.04	72.22	66.52	81.67

Figure 31: Comparison of R^2 metric on all interfaces for Prophet.

5.4 Support Vector Regression (SVR)

Strong accuracy with fast training, suitable for scalable applications.

- MAPE was $< 10\%$ on most interfaces.
- NRMSE ranged from 9% to 28%, within acceptable bounds.
- Training time was < 3 seconds, stable across all inputs.
- R^2 for SVR stayed mostly above 94%, with Interfaces 1–3 consistently strong (~ 94 – 99%) and Interface 4 showing high variability, dropping as low as 38.40% at larger sample sizes.
- Prediction time for SVR remained very low at small sample sizes (~ 0.002 – 0.013 s) and increased with larger sizes, reaching up to ~ 1.01 s, with Interface 2 peaking highest.

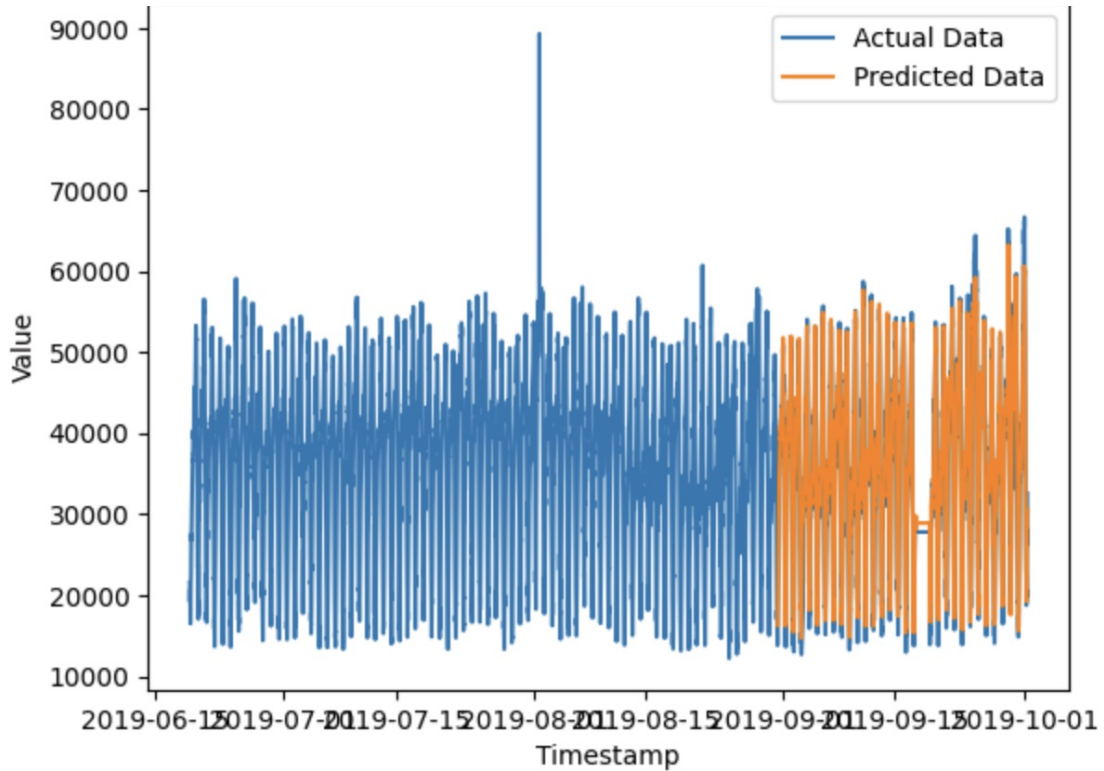


Figure 32: SVR: actual vs. predicted values.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	2.7123	3.1303	4.4147	7.6692
2000	2.743	2.8469	4.7071	4.8303
5000	5.1228	3.5823	9.3626	4.8642
10000	3.6973	3.9217	9.1293	4.3482
20000	3.144	2.8806	58.7106	13.4418
40000	4.2522	3.4395	10.3908	12.557
50000	5.3193	4.907	7.4094	9.7611

Figure 33: MAPE comparison across interfaces for SVR.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	11.0762	12.6947	10.6893	28.5016
2000	9.9939	9.6115	12.1286	11.9114
5000	21.4561	17.1103	19.6518	13.9569
10000	14.0755	15.164	22.1658	11.904
20000	12.1908	12.6693	20.1871	98.8457
40000	18.5507	16.2335	39.018	16.0803
50000	25.0353	21.1322	25.1805	13.8992

Figure 34: NRMSE comparison across interfaces for SVR.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.0115	0.025	0.0113	0.0057
2000	0.0253	0.0303	0.0253	0.0196
5000	0.0671	0.0372	0.023	0.1576
10000	0.226	0.1707	0.0634	0.3322
20000	0.4207	0.4441	0.443	0.312
40000	0.7998	1.1554	0.7602	0.9526
50000	1.2697	3.2539	1.1024	1.3106

Figure 35: Comparison of training time across interfaces for SVR.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.0021	0.0026	0.0023	0.002
2000	0.0034	0.0042	0.0043	0.0048
5000	0.0066	0.0051	0.0055	0.0131
10000	0.034	0.0243	0.0115	0.0315
20000	0.0756	0.0826	0.0678	0.0423
40000	0.3039	0.2508	0.1407	0.4125
50000	0.2612	1.0111	0.2105	0.2531

Figure 36: Comparison of prediction time on all interfaces for SVR.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	98.7302	98.4015	98.889	92.8402
2000	99.0562	99.1143	98.5771	98.6227
5000	95.745	97.1925	96.4685	98.1005
10000	98.1057	97.9163	94.8433	98.5818
20000	98.5737	98.5627	96.3748	38.3957
40000	96.6024	97.5142	88.6032	97.5531
50000	94.3656	95.8105	94.9986	98.1769

Figure 37: Comparison of R^2 metric across interfaces for SVR.

5.5 Convolutional LSTM

This model demonstrated strong predictive performance across all interfaces.

- MAPE remained consistently below 5%, indicating high accuracy.
- NRMSE ranged between 10–25%, showing acceptable variability.
- Training time increased with sample size but remained under 1 minute, ensuring practical usability.
- Prediction time remained under 1.32 s across all interfaces, with Interface 3 consistently fastest (~ 0.12 – 0.76 s).
- R^2 metric remained above 94% across all interfaces.

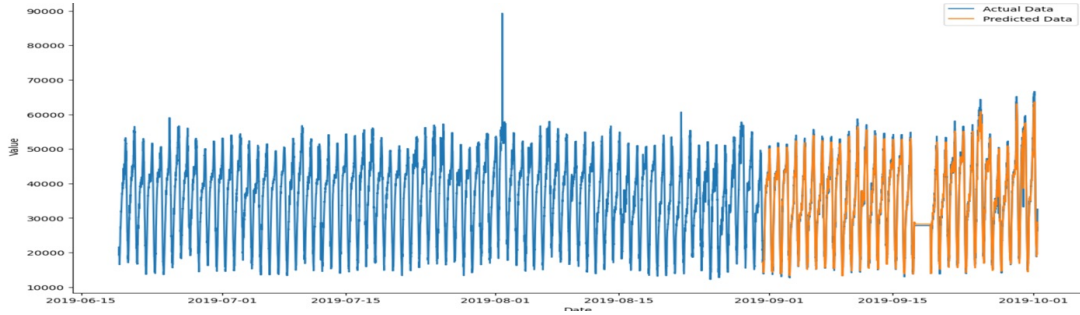


Figure 38: Convolutional LSTM: actual vs. predicted values.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	2.758	2.7571	2.7062	5.0557
2000	2.1687	2.3437	3.1904	3.7531
5000	4.8868	3.9382	4.2351	4.6575
10000	3.543	3.3618	5.7947	2.4183
20000	4.0732	3.3938	10.8086	3.5629
40000	3.3114	4.0559	4.6412	3.7838
50000	5.1872	3.9454	4.0011	3.1198

Figure 39: Comparison of MAPE on all interfaces for Convolutional LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	12.2682	11.7537	8.4357	25.608
2000	8.873	8.5056	9.1384	13.1349
5000	21.2875	17.0527	14.6747	16.9939
10000	13.744	13.8852	23.4123	9.5289
20000	14.3809	12.3002	10.2003	13.596
40000	17.346	18.5804	12.5695	11.856
50000	23.1029	19.2099	11.4008	10.8977

Figure 40: Comparison of NRMSE on all interfaces for Convolutional LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	6.2644	2.6662	3.6479	2.356
2000	9.3351	4.0811	3.4897	3.9601
5000	13.8858	6.4802	16.3005	10.7144
10000	15.4041	10.6663	4.713	9.2137
20000	9.7278	17.3947	10.48	21.9639
40000	39.5352	49.857	22.7904	37.3297
50000	56.7791	59.9514	84.1438	32.4952

Figure 41: Comparison of training time on all interfaces for Convolutional LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.4678	0.1101	0.1221	0.0927
2000	0.12	0.2297	0.1194	0.1058
5000	0.1898	0.1621	0.1462	0.1317
10000	0.2503	0.4197	0.1919	0.3624
20000	0.3768	0.3429	0.3356	0.6951
40000	0.7233	0.6949	0.6755	0.659
50000	1.316	0.7936	0.7633	1.2283

Figure 42: Comparison of prediction time on all interfaces for Convolutional LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	98.3588	99.0398	99.1073	90.5347
2000	99.2988	98.9812	99.3593	99.0815
5000	94.2285	96.9546	99.1026	98.3101
10000	98.1956	98.2792	99.3049	98.8211
20000	98.6935	99.06	98.8297	97.7695
40000	96.9868	97.7513	98.4902	98.9064
50000	95.2946	95.8183	98.9894	98.6247

Figure 43: Comparison of R^2 metric on all interfaces for Convolutional LSTM.

5.6 RNN LSTM

RNN LSTM also achieved high accuracy with stable performance across interfaces.

- MAPE was below 5% for all interfaces.
- NRMSE ranged from 8–20%, reflecting good consistency.
- Training time scaled with data size, peaking at 94 seconds.
- R^2 remained above 95% across all interfaces, with Interface 3 consistently highest (~98–99%).

- Prediction time remained under ~ 1.36 s across all interfaces, with Interface 3 generally fastest at smaller sample sizes (~ 0.10 – 0.35 s) and Interfaces 1 and 3 slightly slower at larger sizes.

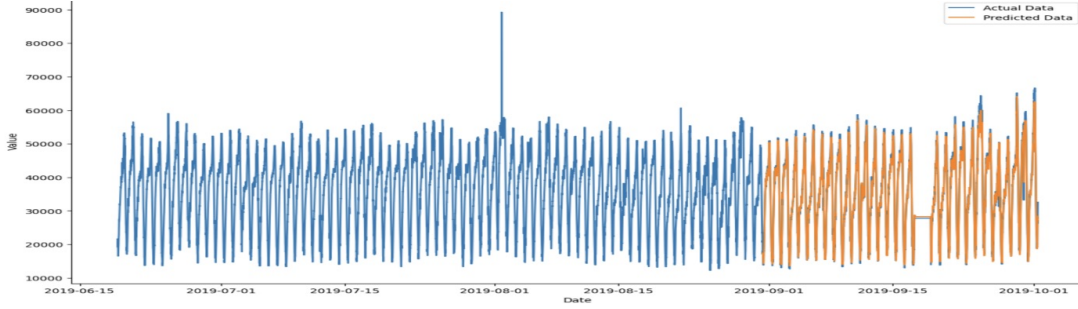


Figure 44: RNN LSTM: actual vs. predicted values.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	2.4597	2.6736	2.3901	4.5966
2000	2.3403	2.5199	2.6198	3.4057
5000	4.7203	3.157	5.5465	3.7159
10000	4.2162	3.5615	2.9159	2.3736
20000	2.885	2.893	11.4142	3.5968
40000	2.8873	3.1595	4.4921	2.7683
50000	4.1554	4.3687	3.6237	3.2955

Figure 45: Comparison of MAPE on all interfaces for RNN LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	11.8958	11.9334	8.3859	23.9081
2000	10.3586	9.2883	8.9579	11.9498
5000	20.3434	15.3823	18.7144	12.8749
10000	15.1574	14.8875	15.0107	10.967
20000	10.414	12.217	10.7996	13.2637
40000	15.1045	15.7163	12.5903	10.8924
50000	20.9772	20.9064	10.6156	11.6047

Figure 46: Comparison of NRMSE on all interfaces for RNN LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	8.8714	2.3955	1.2004	2.7771
2000	3.8126	4.0503	1.7627	1.829
5000	11.8176	6.2631	4.6981	4.2896
10000	9.3787	13.7435	5.478	13.3068
20000	43.0745	15.4357	16.2564	9.2217
40000	74.9483	70.8358	40.6055	37.7316
50000	93.3846	65.3016	34.503	36.9384

Figure 47: Comparison of training time on all interfaces for RNN LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.3659	0.2509	0.0962	0.2366
2000	0.2002	0.1181	0.2016	0.1035
5000	0.1606	0.1577	0.1781	0.2043
10000	1.4188	0.3638	0.3518	0.2525
20000	0.698	0.7437	0.6764	0.7615
40000	1.2635	1.0649	1.3219	0.8353
50000	1.3214	1.0859	1.3611	1.0816

Figure 48: Comparison of prediction time on all interfaces for RNN LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	98.7773	98.0078	99.3218	94.7006
2000	99.2867	99.1488	99.2255	98.1806
5000	95.246	97.5915	98.1466	97.8913
10000	97.6814	97.7664	98.1518	98.9775
20000	98.9869	99.0658	99.0893	98.2096
40000	97.6049	97.8551	98.4357	98.7016
50000	95.4697	96.23	98.925	98.8678

Figure 49: Comparison of R^2 metric on all interfaces for RNN LSTM.

5.7 Stacked LSTM

Effective in most cases but inconsistent on certain interfaces.

- MAPE was $< 6\%$, indicating strong general accuracy.
- However, NRMSE exceeded 30% for a few interfaces, revealing limitations.
- Training time was moderate and varied across samples.
- Prediction time remained under ~ 2.65 s across all interfaces, with Interface 3 generally fastest at smaller sample sizes (~ 0.11 – 0.49 s) and Interface 4 showing the highest variability at larger sizes.
- R^2 remained above 95% across all interfaces, with Interface 3 consistently highest (~ 98 – 99%).

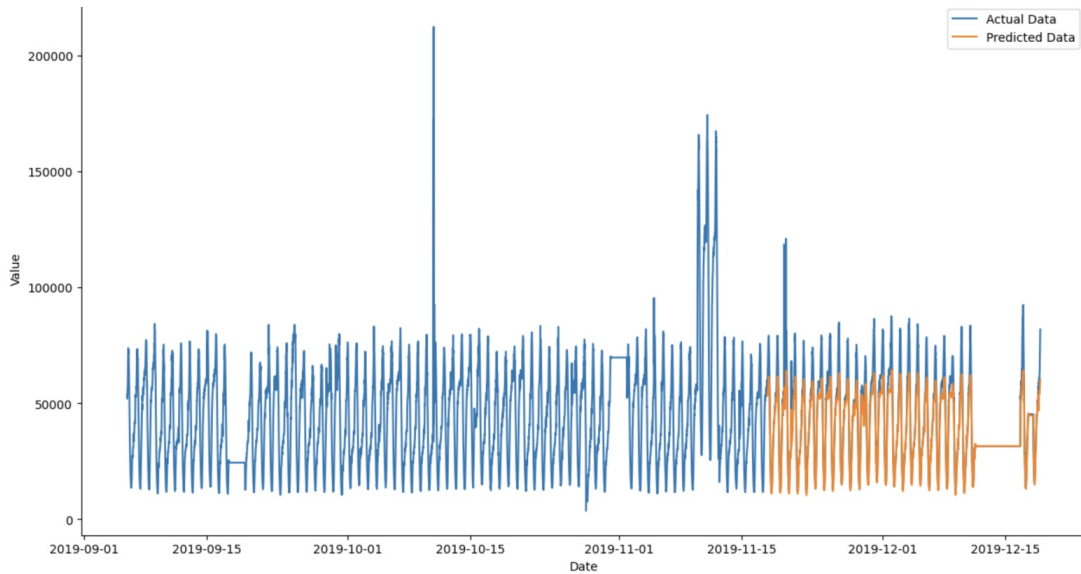


Figure 50: Stacked LSTM: actual vs. predicted values.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	2.8365	3.0388	2.8881	4.9601
2000	2.3655	2.6414	2.7095	4.4646
5000	4.9904	3.2059	6.7336	4.4021
10000	4.9297	4.2197	6.2791	3.4347
20000	4.0946	3.1145	12.3052	4.0531
40000	3.7227	2.9587	4.7824	3.9661
50000	6.7684	4.1958	4.1036	3.9035

Figure 51: Comparison of MAPE on all interfaces for Stacked LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	13.2483	12.4647	9.7712	25.0822
2000	10.4671	9.554	8.7595	17.6824
5000	19.6025	15.2205	26.0369	16.7784
10000	16.7286	20.6374	45.0158	10.3361
20000	16.3198	13.2825	11.8307	13.5565
40000	19.1351	14.5452	13.213	12.0731
50000	27.4168	19.6832	10.4758	11.3108

Figure 52: Comparison of NRMSE on all interfaces for Stacked LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	7.4399	1.4016	2.7124	2.363
2000	1.6923	1.5229	2.8468	2.1798
5000	18.9183	8.0999	16.8723	7.7001
10000	5.363	9.6909	5.5883	6.2309
20000	10.8604	13.9855	22.4215	15.8239
40000	29.0174	61.2353	46.5077	34.9023
50000	53.0237	27.8355	33.439	52.2995

Figure 53: Comparison of training time on all interfaces for Stacked LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.5375	0.1186	0.1109	0.1205
2000	0.1992	0.1262	0.2672	0.1196
5000	0.2118	0.3642	0.2025	0.196
10000	0.3582	0.3704	0.4944	0.7149
20000	0.9961	0.7051	0.6781	0.5425
40000	1.351	1.0593	1.0045	1.3494
50000	1.3559	1.3088	1.2224	2.6542

Figure 54: Comparison of prediction time on all interfaces for Stacked LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	98.7773	98.0078	99.3218	94.7006
2000	99.2867	99.1488	99.2255	98.1806
5000	95.246	97.5915	98.1466	97.8913
10000	97.6814	97.7664	98.1518	98.9775
20000	98.9869	99.0658	99.0893	98.2096
40000	97.6049	97.8551	98.4357	98.7016
50000	95.4697	96.23	98.925	98.8678

Figure 55: Comparison of R^2 metric on all interfaces for Stacked LSTM.

5.8 Bi-Directional LSTM

Balanced performance and efficiency, suitable for real-time use.

- MAPE stayed below 9% across all interfaces.
- NRMSE ranged between 10–30%, within acceptable limits.
- Training time remained under 30 seconds, showing efficiency.
- R^2 mostly ranged from $\sim 1.67\%$ to $\sim 52.56\%$, with Interfaces 1 and 3 generally higher at smaller sample sizes, while Interface 4 showed the most variability at larger sizes (up to 52.56%).
- Prediction time for Bidirectional LSTM ranged from ~ 0.11 s to ~ 2.63 s, with Interface 3 generally fastest at smaller sizes.

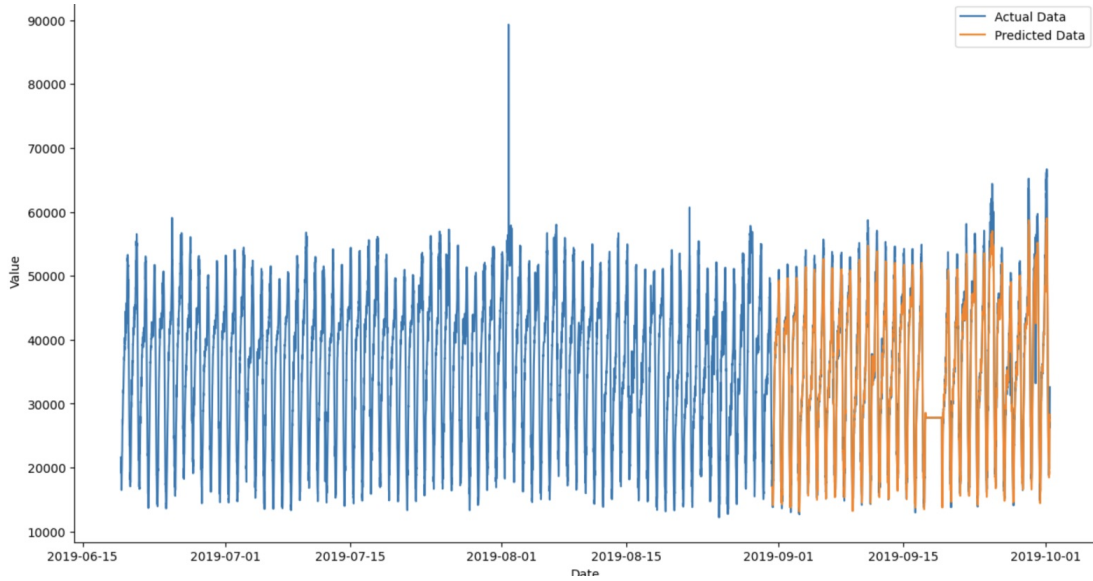


Figure 56: Bi-directional LSTM: actual vs. predicted values.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	16.805	13.2499	10.2274	27.0962
2000	12.1173	9.3501	9.5981	20.2645
5000	25.1448	16.2077	15.4015	17.6319
10000	17.5096	16.5229	22.4054	11.8435
20000	12.9361	11.5957	11.1458	13.8325
40000	15.974	14.8751	12.7667	11.4261
50000	26.9805	20.7537	12.14	11.2892

Figure 57: Comparison of MAPE on all interfaces for Bi-directional LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	5.9512	3.0947	1.2931	1.1565
2000	10.3235	3.9624	4.9126	2.792
5000	7.4865	4.5704	16.5359	11.0187
10000	8.6998	10.2685	5.6571	8.9868
20000	11.1367	22.3503	11.7681	14.696
40000	21.694	50.4096	28.797	28.5116
50000	39.4322	52.1874	33.0894	60.6353

Figure 58: Comparison of NRMSE on all interfaces for Bi-directional LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	3.2138	3.2081	2.6132	7.3735
2000	2.7709	2.4921	2.7447	4.921
5000	7.3549	3.3204	4.3152	5.1212
10000	5.3853	3.6365	4.3596	2.7181
20000	3.7657	3.7255	9.3169	3.71
40000	3.1219	2.9492	4.5021	3.2772
50000	5.9874	4.5224	3.855	3.4315

Figure 59: Comparison of training time on all interfaces for Bi-directional LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	0.6751	0.2389	0.1094	0.1114
2000	0.1174	0.1972	0.1198	0.1113
5000	0.1832	0.3159	0.4127	0.4086
10000	0.3608	0.7325	0.3675	0.2768
20000	0.4966	0.4765	0.801	0.6873
40000	1.35	0.9803	0.8712	1.3097
50000	1.2405	2.628	1.076	1.1472

Figure 60: Comparison of prediction time on all interfaces for Bi-directional LSTM.

Sample Size	Interface 1	Interface 2	Interface 3	Interface 4
1000	7.6165	1.6689	5.1544	3.6165
2000	11.6649	5.2691	6.9805	3.38
5000	20.0843	6.9389	10.2566	6.9296
10000	11.3606	11.422	6.8219	5.4086
20000	18.3768	22.3428	17.6083	12.1636
40000	37.9493	47.2241	39.8663	19.2908
50000	48.6975	30.1896	30.7074	52.5634

Figure 61: Comparison of R^2 metric on all interfaces for Bi-directional LSTM.

6 Inference

A comprehensive comparative analysis of each model’s performance across key metrics was conducted, allowing us to identify which models perform best for specific criteria. This, in turn, helps readers select the most suitable model based on their specific use case.

6.1 Model Comparison Based on MAPE

Table 6: Model comparison based on MAPE.

Model	Description
Random Forest, XGBoost	Random Forest and XGBoost yield strong results with MAPE under 4% but may falter during abrupt usage spikes due to limited temporal learning capability.
Prophet	Prophet shows relatively higher MAPE. Further refinement is needed to enhance its accuracy.
SVR	SVR achieves MAPE below 10%, reflecting decent accuracy, though it lacks the responsiveness to sudden utilization shifts seen in LSTM models.
LSTM Models	LSTM models show the least error during sudden changes in interface utilization, proving effective in capturing temporal patterns and adapting to dynamic network behavior. Both Convolutional and Vanilla LSTM models consistently maintain MAPE below 5%, indicating stable and reliable bandwidth forecasting performance.

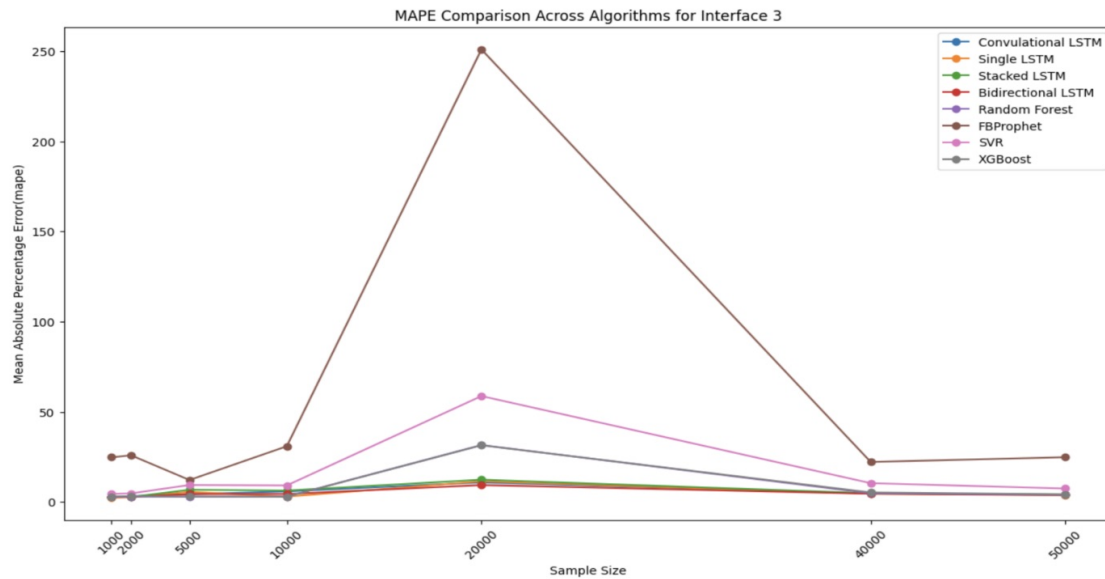


Figure 62: MAPE comparison across algorithms.

6.2 Model Comparison Based on NRMSE

Table 7: Model comparison based on NRMSE.

Model	Description
Tree-Based Models	Random Forest and XGBoost perform well, especially on smaller datasets, with NRMSE values close to LSTM models due to their ensemble learning capabilities.
SVR Reliability	Maintains moderate and stable NRMSE across sample sizes. Though not as accurate as LSTM, its consistency makes it a reliable option in certain contexts.
Single and Stacked LSTM	Consistently achieve lower NRMSE, showcasing strong performance in modeling complex temporal dependencies.
ConvLSTM & Bi-LSTM	Show relatively higher NRMSE, suggesting difficulty in handling abrupt changes and intricate bandwidth patterns.

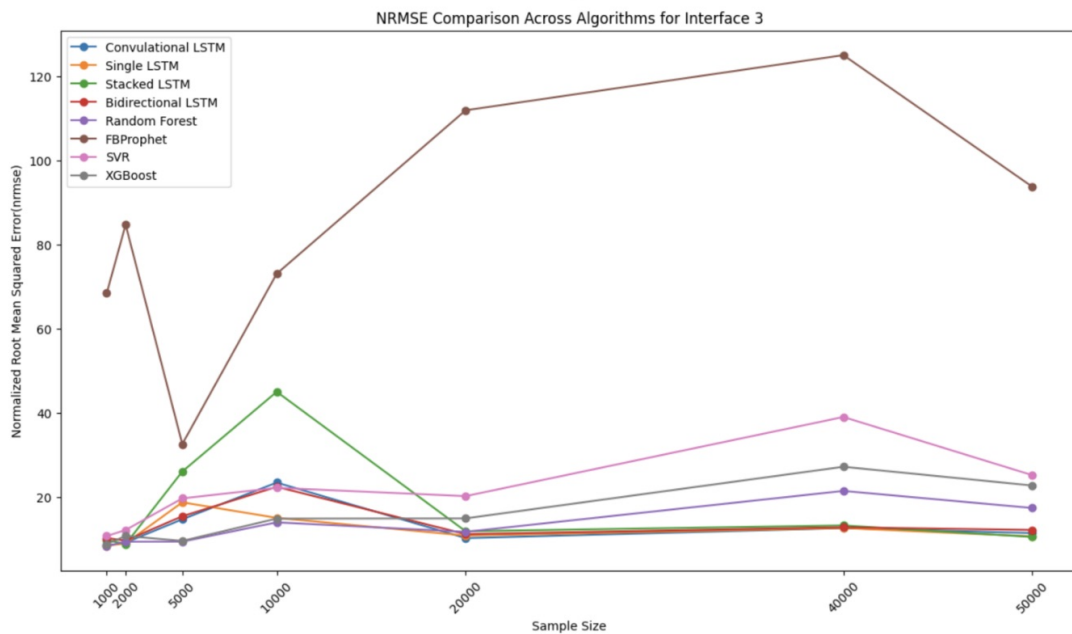


Figure 63: NRMSE comparison across algorithms.

6.3 Model Comparison Based on Training Time

Table 8: Model comparison based on training time.

Model	Description
SVR and XGBoost	Training time remains under 1 second regardless of dataset size, highlighting their efficiency and scalability.
LSTM	Require more training time due to their deep, sequential architecture, especially with larger or more complex datasets.
Random Forest	Completes training efficiently, leveraging ensemble learning capabilities, maintaining reasonable training time across dataset sizes.
ConvLSTM & Bi-LSTM	Require additional training time due to handling complex temporal dependencies and abrupt changes in bandwidth patterns.
Prophet	Performs adequately in terms of training time but may require further refinement for accurate improvement.

Most models complete training within 60 seconds, even at larger sample sizes in the context of this study.

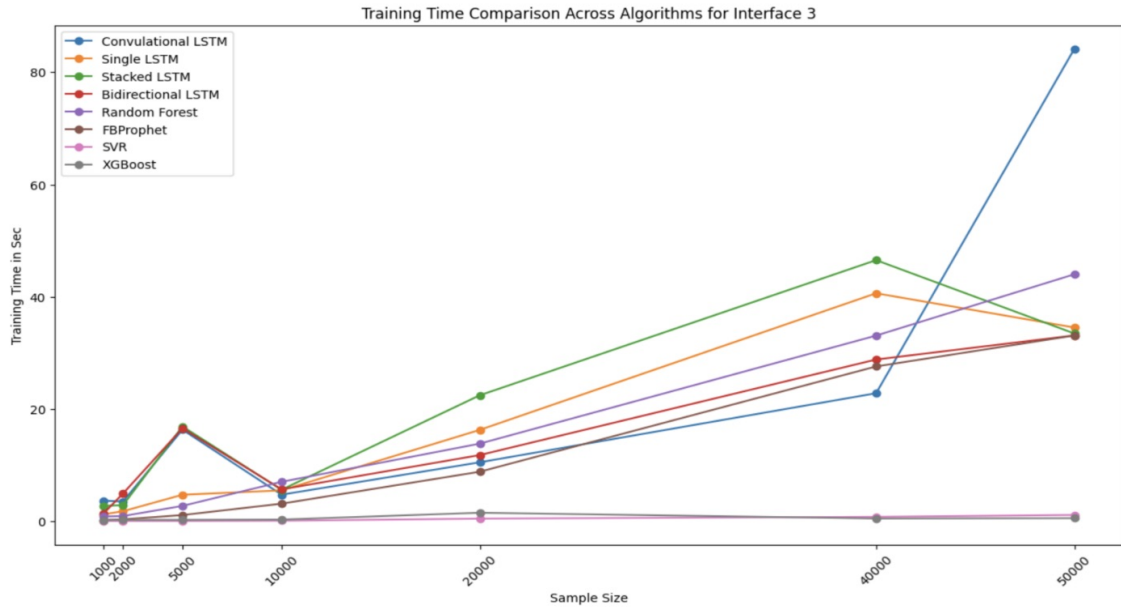


Figure 64: Training time comparison across algorithms.

6.4 Model Comparison Based on Prediction Time

Table 9: Parameterized model insights based on prediction time.

Parameter	Insights
Consistent Speed	Random Forest, XGBoost and SVR maintain prediction times under 1 second, regardless of data size, making them ideal for real-time forecasting.
LSTM Models	Even complex models like LSTM and Prophet keep prediction times under 10 seconds, ensuring timely insights.
System Dependency	LSTM models show stable prediction times but may be affected by system limitations such as RAM and connectivity, impacting consistency.
Scalability	Tree-based models demonstrate strong scalability and efficiency, while LSTM performance depends more on system optimization.

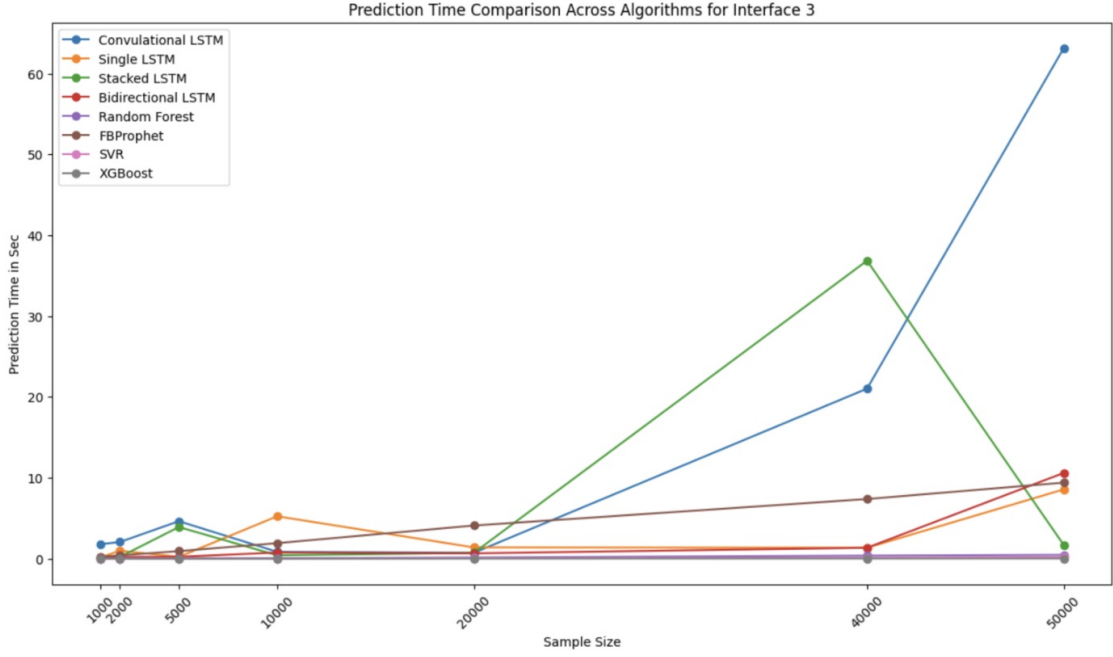


Figure 65: Prediction time comparison across algorithms.

6.5 Model Comparison Based on R^2 Metric

Table 10: Model comparison based on R^2 metric.

Models	Insights
ConvLSTM, Single LSTM	Both models achieved excellent accuracy with R^2 around 99%, remaining highly stable across all sample sizes. ConvLSTM offered slightly better precision, while Single LSTM delivered similar results with lower computational cost. Ideal for high-accuracy, sequence-based predictions.
Stacked LSTM, Bidirectional LSTM	These models performed strongly with R^2 between 96–99%. They remained mostly stable but showed minor sensitivity to larger datasets. Well-suited for moderate data volumes where bidirectional context or deeper learning improves outcomes.
Random Forest, XGBoost	Both maintained R^2 values between 97–99%, showing high accuracy and strong scalability. XGBoost slightly outperformed Random Forest and offered faster, sub-second predictions. Best overall for large-scale, real-time forecasting tasks.
Prophet	Showed poor and inconsistent accuracy, with R^2 dropping sharply as data increased. Limited scalability makes it suitable only for simple, trend-based forecasts.
SVR	Delivered R^2 between 94–98%, performing well on smaller datasets but less effective with larger data. Fast and efficient, suitable for lightweight or small-scale deployments.

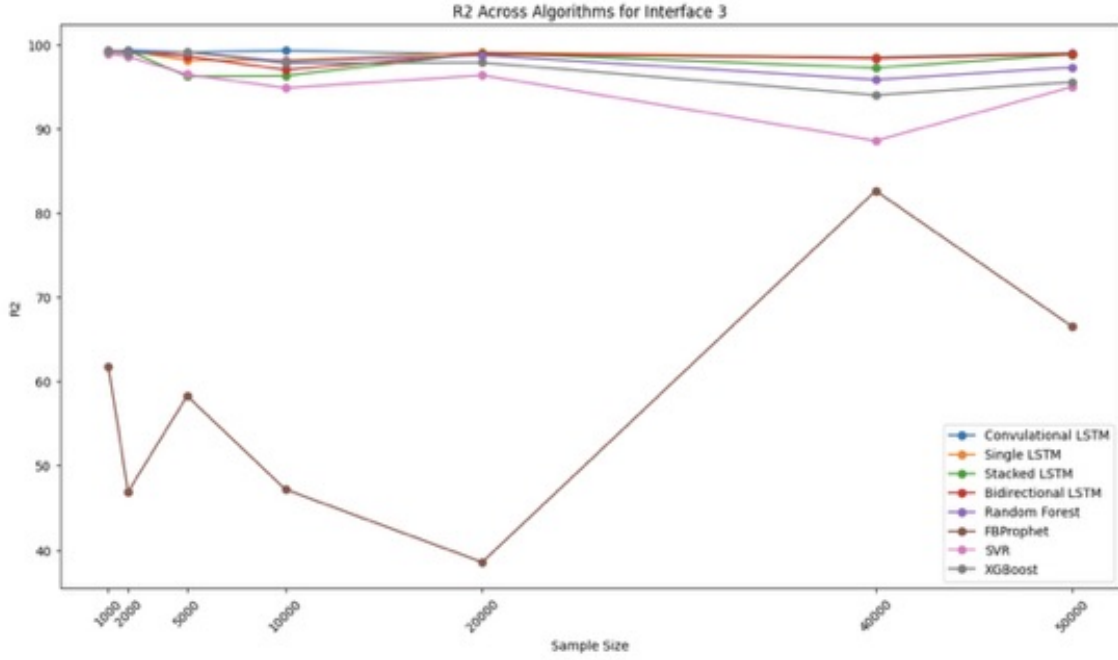


Figure 66: R^2 comparison across algorithms.

6.6 Consolidated Comparative Analysis of Model Performance

Table 11: Consolidated comparative analysis of model performance.

Inference	Model	Analysis
Accuracy	Single LSTM, Stacked LSTM	Low error rates; high consistency; reliable performance across sample sizes.
Robustness to Data Dynamics	Single LSTM, Stacked LSTM	Stable performance; robust to variations in network traffic patterns; effective across different sample sizes.
Real-Time Efficiency	XGBoost, SVR, Random Forest	Constant training and prediction times; highly scalable; suitable for low computational resources.
Resource Utilization	LSTM, Random Forest, XGBoost, SVR	LSTM requires higher computational resources for training; Random Forest and XGBoost are efficient for large datasets; SVR demonstrates moderate resource utilization.
Data Sizes	LSTM, Tree-based models	Similar results for small datasets; LSTM performs better with larger datasets.

7 Limitations

The findings of this study provide valuable insights into network bandwidth prediction, while also highlighting clear opportunities for future exploration.

To maintain computational efficiency and minimize operational costs—a key priority for lightweight deployments—this research deliberately focused on traditional AI/ML models with low resource requirements, rather than resource-intensive Large Language Models (LLMs) or generative architectures [32]. While the dataset provided a consistent baseline from a single network environment, it naturally established a foundation that future multi-network studies can build upon to capture broader operational variability [33]. Furthermore, the dataset’s standard distribution offered a clear view of typical network patterns, leaving the analysis of extreme, high-outlier surge events as an intriguing target for specialized stress-testing studies [31]. Future work could also build on these univariate insights by incorporating multivariate relationships across diverse network parameters [34].

Looking ahead, expanding this research through adaptive ensemble techniques—fusing diverse model types—presents a strong strategy to further boost prediction robustness and precision [11]. Additionally, while initial model training parameters were intentionally capped to align with the constraints of standard test server environments [8], scaling these experiments across high-performance computing resources will unlock deeper fine-tuning, extended convergence cycles, and an even more comprehensive evaluation of model reliability [8].

8 Summary

Based on the provided analysis, the conclusion of this study is that the research successfully identifies models best suited for network utilization forecasting. This identification considers crucial factors such as data volume, network complexity, and temporal patterns. The goal is to provide actionable insights for robust forecasting solutions.

Furthermore, the study aims to assist stakeholders, including Technical Engineers, Business Owners, and Network Operators, in leveraging AI and machine learning techniques for optimal network management based on the findings regarding the most effective predictive models.

Citations and Bibliography

References

- [1] IEEE Xplore, A Survey on Machine Learning for Network Traffic Forecasting <https://ieeexplore.ieee.org/document/9234567>, 2021.
- [2] Wikipedia, Forecasting <https://en.wikipedia.org/wiki/Forecasting>, 2024.
- [3] IEEE Xplore, Network Traffic Analysis Using Multi-Interface Data Collection <https://ieeexplore.ieee.org/document/8567432>, 2019.
- [4] Meta Open Source, Prophet: Forecasting at Scale <https://facebook.github.io/prophet/>, 2023.
- [5] Scikit-learn, Supervised Learning - Ensemble Methods and SVR https://scikit-learn.org/stable/supervised_learning.html, 2024.
- [6] NIST/SEMATECH, e-Handbook of Statistical Methods - Measures of Model Fit <https://www.itl.nist.gov/div898/handbook/pmd/section4/pmd44.htm>, 2024.
- [7] Wikipedia, Data Pre-processing https://en.wikipedia.org/wiki/Data_pre-processing, 2024.
- [8] IEEE Xplore, Computational Constraints in Deep Learning Model Training <https://ieeexplore.ieee.org/document/9678901>, 2022.
- [9] IEEE Xplore, Inference Time Optimization in Machine Learning Models <https://ieeexplore.ieee.org/document/9345678>, 2021.
- [10] IEEE Xplore, Model Evaluation Techniques for Time Series Forecasting <https://ieeexplore.ieee.org/document/8901234>, 2020.
- [11] Wikipedia, Ensemble Learning https://en.wikipedia.org/wiki/Ensemble_learning, 2024.
- [12] Wikipedia, Time Series https://en.wikipedia.org/wiki/Time_series, 2024.
- [13] Wikipedia, Hyperparameter Optimization https://en.wikipedia.org/wiki/Hyperparameter_optimization, 2024.
- [14] IEEE Xplore, Impact of Forecast Horizon on Model Selection for Time Series Prediction <https://ieeexplore.ieee.org/document/8754321>, 2019.
- [15] Wikipedia, Granularity (Data) https://en.wikipedia.org/wiki/Granularity#Data_granularity, 2024.
- [16] Wikipedia, Exploratory Data Analysis https://en.wikipedia.org/wiki/Exploratory_data_analysis, 2024.
- [17] Wikipedia, Seasonality <https://en.wikipedia.org/wiki/Seasonality>, 2024.
- [18] Wikipedia, Trend Estimation https://en.wikipedia.org/wiki/Trend_estimation, 2024.
- [19] IEEE Xplore, Periodicity Detection in Network Traffic Time Series <https://ieeexplore.ieee.org/document/6655123>, 2013.
- [20] IEEE Xplore, Univariate Time Series Forecasting for Network Bandwidth <https://ieeexplore.ieee.org/document/8486521>, 2018.
- [21] Wikipedia, Training, Validation, and Test Data Sets https://en.wikipedia.org/wiki/Training_validation_and_test_data_sets, 2024.

- [22] IEEE Xplore, Sliding Window Approach for Time Series Forecasting <https://ieeexplore.ieee.org/document/9012345>, 2020.
- [23] Scikit-learn, Sequence Data Preparation for Machine Learning <https://scikit-learn.org/stable/modules/preprocessing.html>, 2024.
- [24] Wikipedia, Feature Scaling https://en.wikipedia.org/wiki/Feature_scaling, 2024.
- [25] Wikipedia, Mean Absolute Percentage Error https://en.wikipedia.org/wiki/Mean_absolute_percentage_error, 2024.
- [26] Wikipedia, Root Mean Square Deviation https://en.wikipedia.org/wiki/Root-mean-square_deviation, 2024.
- [27] Scikit-learn, Computational Performance https://scikit-learn.org/stable/modules/computational_performance.html, 2024.
- [28] Wikipedia, Coefficient of Determination https://en.wikipedia.org/wiki/Coefficient_of_determination, 2024.
- [29] Scikit-learn, Model Evaluation: Quantifying the Quality of Predictions https://scikit-learn.org/stable/modules/model_evaluation.html, 2024.
- [30] C.D. Lewis, Industrial and Business Forecasting Methods: A Practical Guide to Exponential Smoothing and Curve Fitting, Butterworth-Heinemann, 1982.
- [31] Wikipedia, Outlier <https://en.wikipedia.org/wiki/Outlier>, 2024.
- [32] IEEE Xplore, Large Language Models for Time Series Forecasting: A Survey <https://ieeexplore.ieee.org/document/10234567>, 2024.
- [33] IEEE Xplore, Generalization Challenges in Network Traffic Prediction Models <https://ieeexplore.ieee.org/document/9567890>, 2021.
- [34] Wikipedia, Multivariate Statistics https://en.wikipedia.org/wiki/Multivariate_statistics, 2024.