International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V73I8P103

Volume 73 Issue 8, 15-24, August 2025
© 2025 Seventh Sense Research Group®

Original Article

AI/ML-Driven IP Multimedia System (IMS) Application Scaling and Auto Tune Config for Telco Networks Operating in Cloud Platforms

Harikishore Allu Balan[1], Bikash Agarwal[2]

[1]*T-Mobile, Principal Solution Architect, WA, USA.*
[2]*T-Mobile, Principal Engineer System Design, WA, USA.*

[1]*Corresponding Author : harikishore.allubalan@ieee.org*

Received: 13 June 2025 Revised: 20 July 2025 Accepted: 13 August 2025 Published: 30 August 2025

Abstract *- This study explores how IP Multimedia Systems (IMS) applications can be optimized and scaled effectively within telecom networks that rely on cloud infrastructure. IMS supports a wide range of services, including Voice over IP (VoIP), Video over IP, and Rich Communication Services (RCS), and utilizes standardized components such as P-CSCF, S-CSCF, I-CSCF, TAS, RCS, and MRF in accordance with 3GPP specifications. A unified method for gathering diverse operational data has been proposed, covering performance indicators from the network, infrastructure usage, application-level behavior, and surrounding environmental factors. From this pool of data, key indicators—like call success and failure rates, scam call detection, and voicemail activity—are extracted and analyzed. To make sense of these trends, several machine learning models, including Random Cut Forest (RCF), XGBoost, and a basic K-Nearest Neighbors (KNN) approach, are used. Their predictive strength is measured using common statistical tools such as R-squared (R²), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). This evaluation helps determine which model is most suitable for anticipating specific challenges such as SIP errors, call drops, latency issues, and network congestion. Based on these predictions, the system can automatically fine-tune its settings to adapt to changing network conditions. By integrating with CI/CD pipelines, these adjustments can happen in near real-time. The end result is a responsive and cost-effective framework for managing IMS resources in cloud-based telecom environments.*

Keywords *- IP Multimedia System (IMS), Voice over IP (VoIP), Video over IP, Rich Communication Services (RCS), Cloud Platforms, Dynamic Scaling, Predictive Analytics, Machine Learning (ML), Random Cut Forest (RCF), XGBoost, K-Nearest Neighbors (KNN), R² (Coefficient of Determination), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), SIP Error Codes, Call Success Rate, Call Drop Rate, Registration Success Rate, Scam Call Detection, Voicemail Frequency, Network Congestion, Latency Forecasting, CI/CD Deployment Pipelines, Adaptive Network Configuration, Automated Resource Management component, formatting, style, styling, insert (key words).*

1. Introduction

This paper focuses on improving the way IP Multimedia System (IMS) applications are scaled and optimized within cloud-based telecom infrastructures. IMS plays a central role in delivering services such as Voice over IP (VoIP), Video over IP, and Rich Communication Services (RCS). It relies on several core network elements—including the Proxy-CSCF, Serving-CSCF, Interrogating-CSCF, Telephony Application Server (TAS), Media Resource Function (MRF), and RCS components—as outlined in the 3GPP standards. Together, these building blocks enable seamless multimedia communication in modern telecommunications systems.

With the growing adoption of cloud-native architectures, telecom providers are shifting toward the use of virtualization tools such as virtual machines and containers. These technologies make it possible to build applications that are not tied to specific hardware, supporting deployment through Continuous Integration and Continuous Deployment (CI/CD) pipelines. This strategy allows systems to be rolled out smoothly across multiple environments, including public cloud services like AWS, Google Cloud, and Azure, as well as private platforms like Red Hat OpenStack, Platform9, and Rancher Kubernetes. By using these flexible platforms, service providers can dynamically adjust resources to meet fluctuating demand, supported by advanced auto-scaling features and AI/ML-driven analytics.

The study takes a closer look at several operational metrics—including call logs, hardware usage stats, alarms, and container resource utilization—to identify meaningful patterns. These data points are transformed into features suitable for building predictive models and spotting anomalies. Machine learning algorithms such as Random Cut Forest (RCF), XGBoost, and K-Nearest Neighbors (KNN) are then applied to forecast potential problems before they impact performance. The insights gained are fed back into automated CI/CD workflows and orchestration tools, enabling swift and intelligent responses that help maintain service stability and ensure continuous availability of IMS functions.

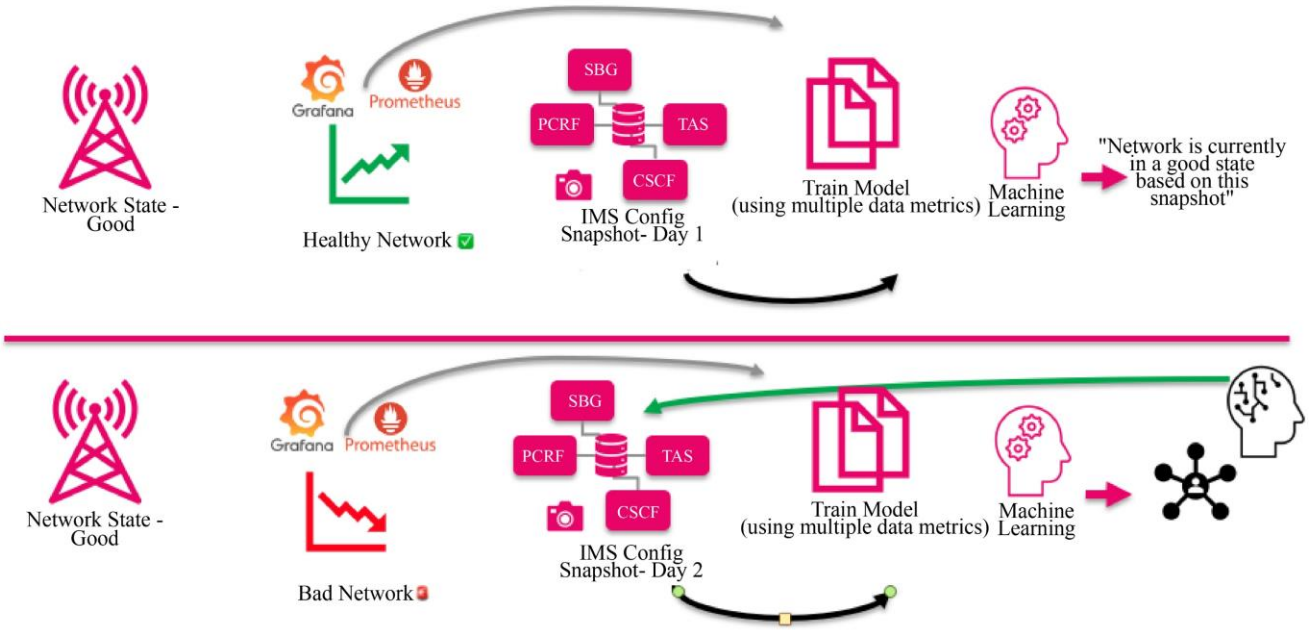
Network State - Good
Grafana
Prometheus
Healthy Network
SBG
PCRF
TAS
CSCF
IMS Config Snapshot- Day 1
Train Model (using multiple data metrics)
Machine Learning
"Network is currently in a good state based on this snapshot"
Network State - Good
Grafana
Prometheus
Bad Network
SBG
PCRF
TAS
CSCF
IMS Config Snapshot- Day 2
Train Model (using multiple data metrics)
Machine Learning

Fig. 1 Network ML feedback Overview

The following Figure illustrates the integrated AI/ML-driven architecture employed for real-time monitoring, analytics, and dynamic optimization of IMS network performance:

2. Literature Review

In recent years, the use of AI and machine learning has become increasingly important for monitoring and resolving issues in telecom networks. One notable contribution comes from Bagmar et al. (2025), who applied ensemble learning methods—including Random Forest and XGBoost—to accurately forecast latency and data throughput in environments using multiple cloud platforms.

Their approach incorporated a wide range of metrics that covered not only network activity but also infrastructure conditions, application performance, and environmental influences. Impressively, their models achieved high accuracy, with R^2 values exceeding 0.95.

Building on this, Huang et al. (2019) applied deep learning strategies to improve data throughput in 5G networks. They focused particularly on adjusting precoding methods dynamically in large-scale MIMO systems. This technique enabled better real-time forecasting of network performance.

Nikravsh et al. (2016) explored a range of machine learning models, including variants of Multilayer Perceptrons (MLP and MLPWD) as well as Support Vector Machines (SVM). Their work showed that careful selection of features and algorithms is essential for making short-term predictions about network traffic with high precision.

More recently, Pathak et al. (2024) introduced quantum computing into the mix. They developed a model known as VQR to boost the accuracy of predictions related to resource distribution. The model achieved a very low mean squared error (0.0081), suggesting strong potential for use in scenarios where precise decision-making is essential, such as in 5G infrastructure.

Lastly, one of the researcher highlighted the transformative impact of AI on enterprise-scale multi-cloud environments, particularly for IMS applications. Their findings emphasized enhanced adaptability, proactive resource allocation, and dynamic scalability enabled by AI-driven approaches.

3. Methodology

3.1. Data Collection and Feature Engineering

Data is aggregated from IMS network elements, encompassing metrics from network, infrastructure, applications, and environmental factors. Sourcing data from multiple applications with clearly defined metrics, KPIs, alerts, events, success states, and hardware utilization into a common structured data format is crucial. Ensuring that the data is thoroughly sanitized and devoid of any Personally Identifiable Information (PII) before feature engineering is essential to maintain compliance and data privacy.

The data ingestion pipelines utilized across all applications employ JSON file formats, streaming metrics into event streams consumed via a Kafka pipeline secured through mutual TLS authentication. This unified data collection endpoint ensures comprehensive coverage and consistently high-quality datasets.

3.1.1. Metrics Utilized

- Network Metrics: Signal strength, bandwidth utilization, packet loss rate.
- Infrastructure Metrics: Cloud provider metrics, resource utilization.
- Application Metrics: Session counts, call types, and messaging volume.
- Environmental Metrics: Geolocation data, distance to edge nodes.

Feature extraction methods highlight critical KPIs such as:

- Call Success Rate
- Call Drop Rate
- Registration Success Rate
- Scam Caller Detection
- Calls to Voicemail
- Call Setup Failure Rate

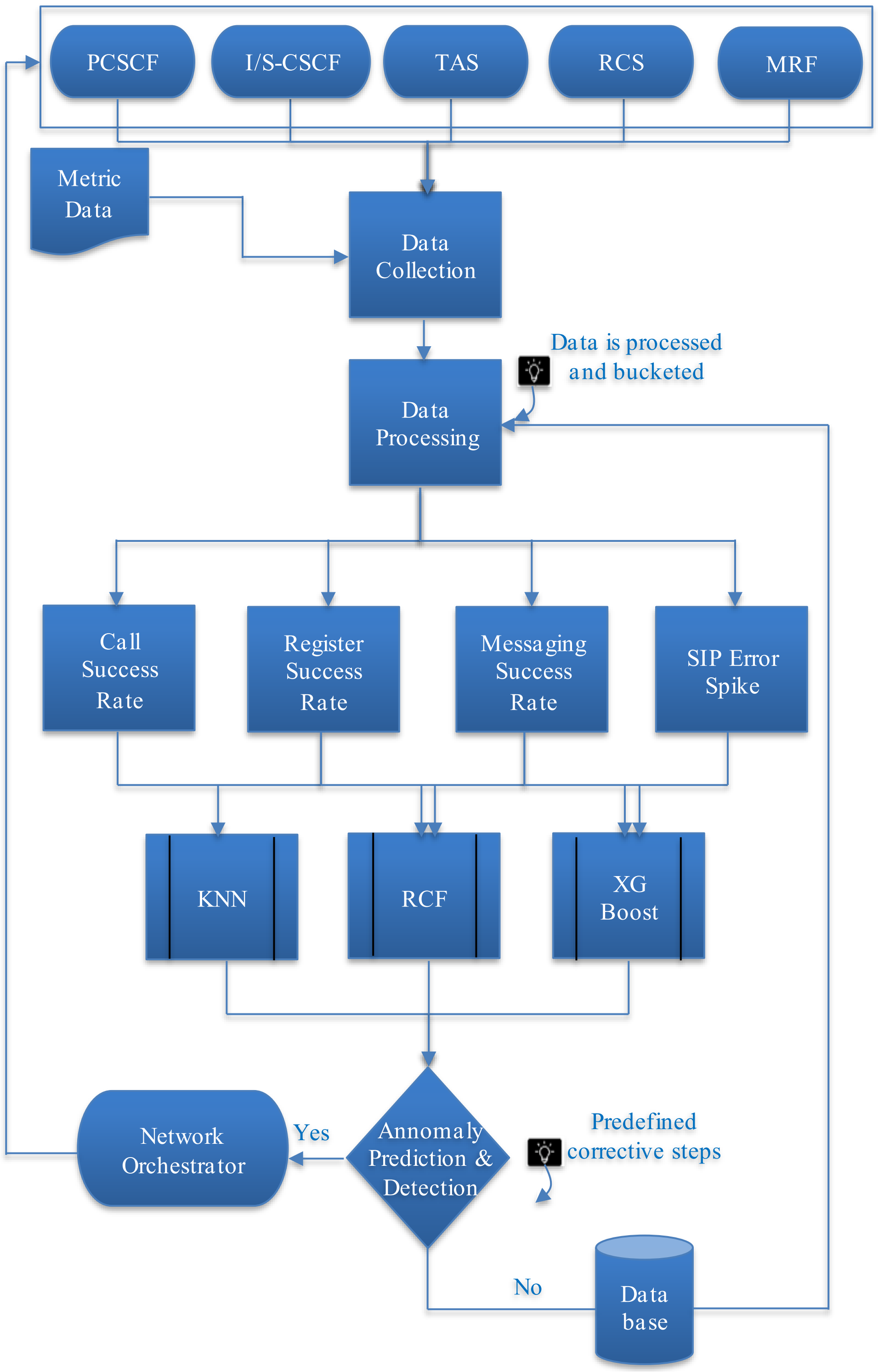

Fig. 1 Data Aggregation and Framing for ML adaptation with a feedback loop

3.2. ML Models and Predictive Analytics

In training the machine learning models, a sliding window approach was implemented to effectively capture time-based relationships in the data. Window sizes were adjusted between 5 and 30 minutes, depending on the specific prediction tasks and intervals required. Each model was intentionally chosen based on its suitability for analyzing distinct subsets of data drawn from various application sources. By focusing on specific types of anomalies, such as unexpected CPU spikes and software crashes, the models underwent rigorous adjustments in their temporal parameters to ensure accurate and consistent tracking of these events across multiple data metrics.

This methodical refinement improved the predictive reliability and effectiveness of the anomaly detection framework. Anomalies, such as CPU spikes and software crashes, were closely monitored, with machine learning models dynamically adjusted through time series analysis to maintain a consistent and reliable stream of event data from diverse metrics. Hyperparameters were tuned using Bayesian optimization alongside a rigorous 5-fold cross-validation process to achieve optimal performance.

3.2.1. k-Nearest Neighbors (kNN) Overview

The k-Nearest Neighbors (KNN) algorithm is widely used for both classification and regression tasks due to its simplicity and adaptability. It operates by comparing a new data point to its ‘k’ closest counterparts in the training dataset, using a distance-based similarity measure to guide classification or prediction. The value of ‘k’ typically ranges from 3 to 10 and is fine-tuned to improve model performance. Distance metrics like Euclidean, Manhattan, and Minkowski are commonly applied, each offering different advantages depending on the structure and distribution of the data.

As an instance-based learning method, KNN does not involve a traditional training phase. Instead, it responds to new inputs by referencing patterns in historical data, making it both easy to implement and interpret. This characteristic makes KNN especially appealing in scenarios where model transparency and explainability are important. It is relatively straightforward to implement and interpret, making it popular for applications requiring transparent and understandable modeling processes.

KNN Euclidean distance formula:

$$\sqrt{\sum_{i=0}^{k}(x_i - y_i)^2} \tag{1}$$

KNN Manhattan distance formula:

$$\sum_{i=1}^{k}|x_i - y_i| \tag{2}$$

KNN Minkowski distance formula:

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{\frac{1}{q}} \tag{3}$$

Where q can be any real value between 0 and 1.

3.2.2. Random Cut Forest (RCF) for Anomaly Detection

Random Cut Forest (RCF) is an unsupervised machine learning algorithm specifically designed to detect anomalies in data streams. It operates by building multiple isolation trees through random partitioning of data, identifying points that require fewer splits as potential anomalies.

The primary utility of RCF is detecting outliers or irregular data points, such as sudden spikes in CPU utilization, software crashes, or unusual patterns in system behavior. While traditionally leveraged for anomaly detection, RCF can also provide valuable insights into potential future failures by identifying early-stage deviations that precede significant system issues. This makes RCF particularly useful in proactively managing and mitigating network anomalies before they escalate into critical failures.

The method that uses a group of interconnected classifiers to generate decision trees. These classifiers are really just individual learners put together. A greedy approach is used to generate N decision trees from a training subset. In equation (4), they can see that the decision trees are combined to make a majority-vote forecast for each class, denoted as y_i, and the corresponding probability, p_n(y_i)

$$R_{(yi)} = \frac{1}{N}\sum_{n=1}^{N} p_n(y_i) \tag{4}$$

The hyperparameters used in the RF model are n_estimators=150 (the number of trees to be generated) and random_state=100 (for reproducibility and ensuring the same splits in the data).

3.2.3. XGBoost

XGBoost (short for eXtreme Gradient Boosting) is a highly efficient and accurate machine learning algorithm that has become a go-to choice for many predictive modeling tasks. It functions by gradually improving its predictions through the addition of decision trees—each new tree focuses on correcting the mistakes made by previous ones. This sequential learning strategy allows the model to reduce error over time.

What sets XGBoost apart is its built-in regularization mechanism, which helps prevent overfitting and ensures that the model performs well even on new data. It also supports parallel processing, making it suitable for large datasets where computational speed is essential. Due to its ability to handle complex data relationships, XGBoost is particularly effective in telecom environments where multiple variables interact in nonlinear ways. Mathematically the model can be represented as an equation. (5)

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \tag{5}$$

Where:

- $\hat{y}_i$ is the final predicted value for the ith data point
- K is the number of trees in the ensemble
- $f_k(x_i)$ represents the prediction of the K^th tree for the i^th data point.

For the XGBoost algorithm, the optimization goal can be written as

$$Obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \quad (6)$$

Where l is the training loss function, $\hat{y}_i$ is the prediction for the ith instance, and Ω represents the regularization term to control model complexity.

4. Model Evaluation

To evaluate how well the models performed, three commonly used statistical metrics were applied: R-squared (R^2), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These indicators help quantify both the accuracy and reliability of the predictions generated by each algorithm.

4.1. R-Square

The R^2 (R-squared value), also known as the coefficient of determination, measures how well the independent variables in a model explain the variance observed in the dependent variable. It ranges between 0 and 1, where a score closer to 1 means the model's predictions align closely with actual outcomes. A higher R^2 generally reflects stronger predictive performance.. The value may be anything from 0 to 1, with 1 indicating an ideal match and 0 indicating no explanatory power. Higher R^2 values indicate better model performance. The R^2 is calculated as formula (7):

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \quad (7)$$

Where:

- y_i is an actual value
- $\hat{y}_i$ is the forecasted value
- $\bar{y}$ is a means of actual values

4.2. Mean Squared Error (MSE)

The MSE is a widely utilized statistic to evaluate the accuracy of a regression model. By squaring the discrepancies between actual observations and the outcomes predicted by the regression ML model, this metric determines the average. The MSE is calculated as formula (8):

$$MSE = \frac{1}{m}\sum(y_{pred} - y_{ture})^2 \quad (8)$$

Where:

- y_{pred} is the forecasted output,
- y_{ture} is the real observation,
- Σ is the number of observations
  - m is the sum of all observations.

4.3. Root Mean Square Error (RMSE)

The RMSE calculated how far the data points were from the regression line. It is involved in the validation of predicting equations used in forecasting, climatology, and regression analysis, as well as measuring the spread of residuals. The RMSE is calculated as formula (9):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}} \quad (9)$$

Where $'i'$ corresponds to a single variable in each of the relevant columns, $'N'$ is equivalent to the number of complete data points, x_i. This means actual observation of time series data and $\hat{x}_i$ means estimated time series.

These machine learning models are trained specifically to anticipate significant performance issues such as SIP error codes and network congestion events. By comprehensively understanding the entire network call flow, the models effectively trace the root causes and origins of any observed anomalies, correlating these with specific software versions. The insights derived from this analysis are directly fed back into CI/CD pipelines, enabling automated tuning of configurations. In cases where the system detects elevated CPU usage, Kubernetes schedulers automatically scale the applications to handle increased demand efficiently. This proactive approach swiftly addresses network anomalies and optimizes performance and resource utilization, significantly reducing the need for manual oversight.

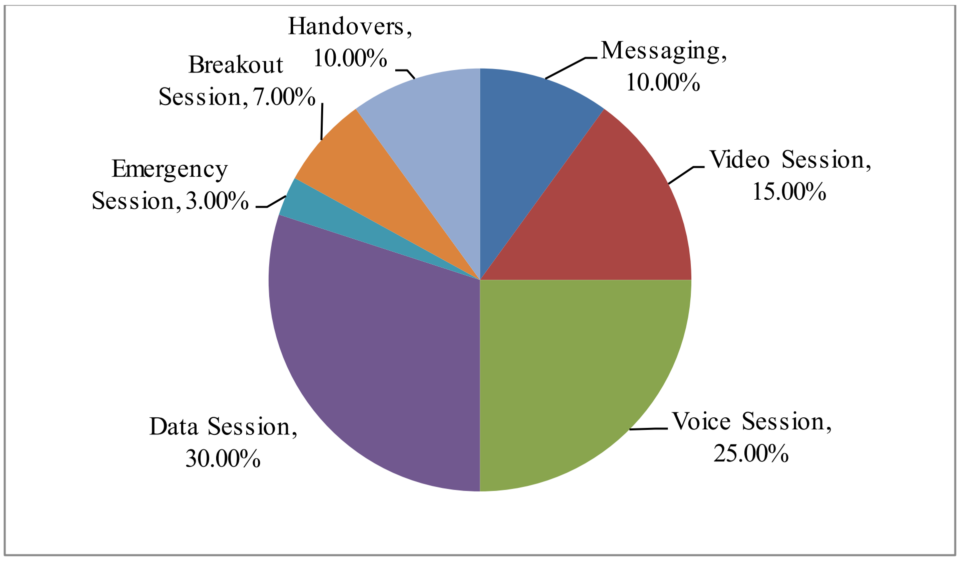

Fig. 3 User Traffic Pattern Metrics Distribution

5. Results and Analysis

The applied ML models exhibit high predictive accuracy, with XG Boost and RCF significantly outperforming baseline models, particularly in anomaly detection and predictive accuracy for latency and congestion events. The experimental setup involves collecting extensive data patterns from one million users. These data include sessions categorized as data sessions, voice sessions, video sessions, messaging activities, handovers, breakout sessions, and emergency sessions. This detailed data collection provides a comprehensive foundation for robust predictive analytics.

The distribution of user traffic pattern metrics feeding into the data pipeline is illustrated in the pie chart above, showcasing the proportionate representation of each session type in the dataset.

5.1. Experimental Setup

The outcomes of this study highlight the superior performance of the XG Boost model compared to the KNN and RCF models in predicting IMS network election resource allocation, utilization and error prediction, with the XG Boost model achieving higher R-squared and lower MSE and RMSE. The system was run on a locally installed server environment with a 64-core server with 128 GB of RAM, with a GTX 1660 Ti GPU installed, hosting Red Hat Host OS 7 and using Platform 9 Kubernetes cluster. The data collection and aggregation were done using the application Kafka Streams in JSON format and stored in S3 storage buckets. Exploratory data analysis and building regression models for IMS network application performance optimization were accomplished using Pandas, NumPy, Matplotlib, Seaborn, and scikit-learn libraries.

5.1.1. Case 1: Call Success Rate

This section compares the performance of Gradient Boosting Regressor, Random Cut Forest and K-Nearest Neighbors (KNN) for predicting IMS application call handing, as shown in Table I. The Gradient Boosting Regressor forms multiple decision trees to formulate the model and gives robust results, but it needs proper tuning. However, the simpler XG Boost model does better in this work; it attains higher R-squared, MSE, and RMSE, making it efficient in IMS call success rate prediction with traffic pattern in the allocated application resources.

Table 1. Call Success rate prediction across ML Models for a 24-hour duration

MODELS	KNN	Random Forest Regressor	XG Boost
R^2-Score	0.923	0.945	0.958
MSE	0.0059	0.0034	0.0029
RMSE	0.0858	0.0580	0.0403

5.1.2. Case 2: SIP Error prediction for 480 and 503

The experimental setup for this SIP error trend uses 2 weeks of user call model, where a busy hour pattern of 10000 calls was used to introduce known SIP error failures to fine-tune the model and then later use the ML models to detect abnormalities and adjust to new error trends.

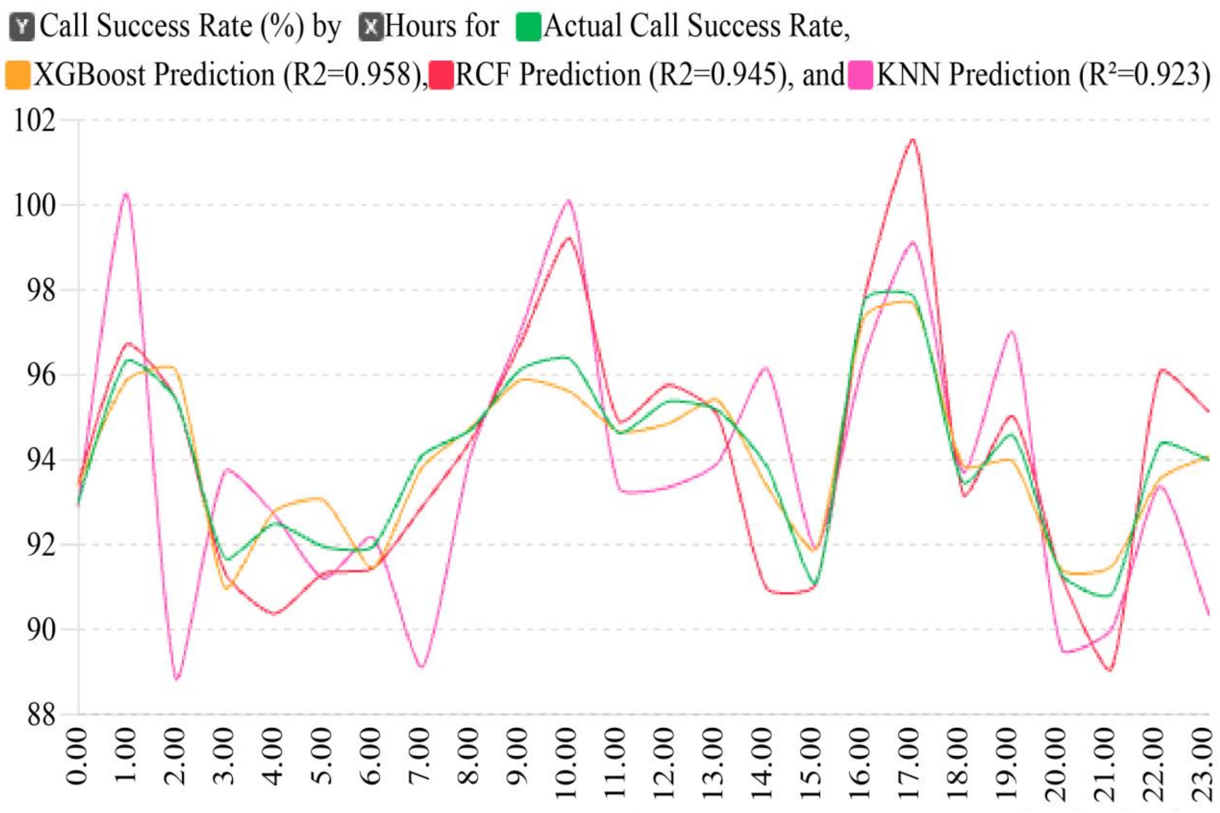

Fig. 4 Call Success Rate analysis and prediction by ML Models

Table 2. SIP Error rate prediction across ML Models for 14 days duration

SIP Error Code	Models	R² Score	MSE	RMSE
480	KNN	0.910	0.0065	0.0806
480	Random Forest	0.937	0.0039	0.0624
480	XGBoost	0.952	0.0030	0.0548
503	KNN	0.905	0.0070	0.0837
503	Random Forest	0.930	0.0041	0.0640
503	XGBoost	0.949	0.0031	0.0557

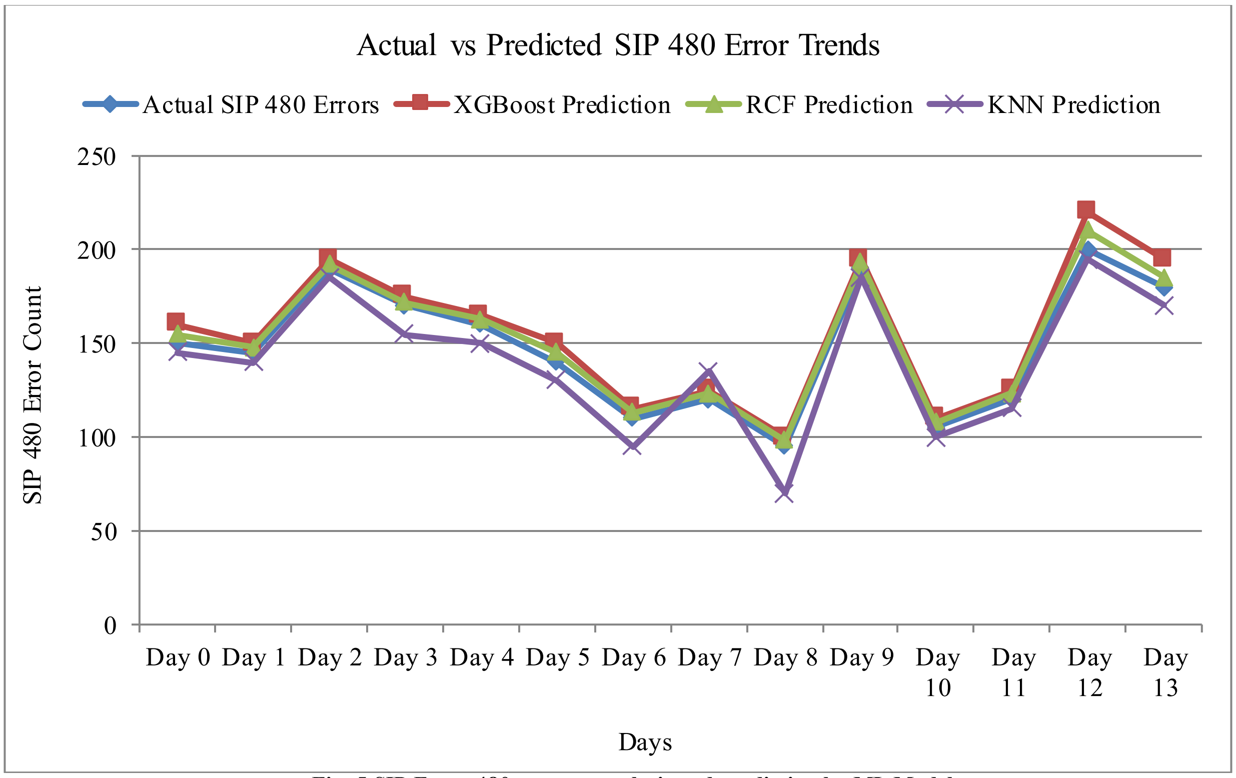

Fig. 5 SIP Error 480 pattern analysis and prediction by ML Models

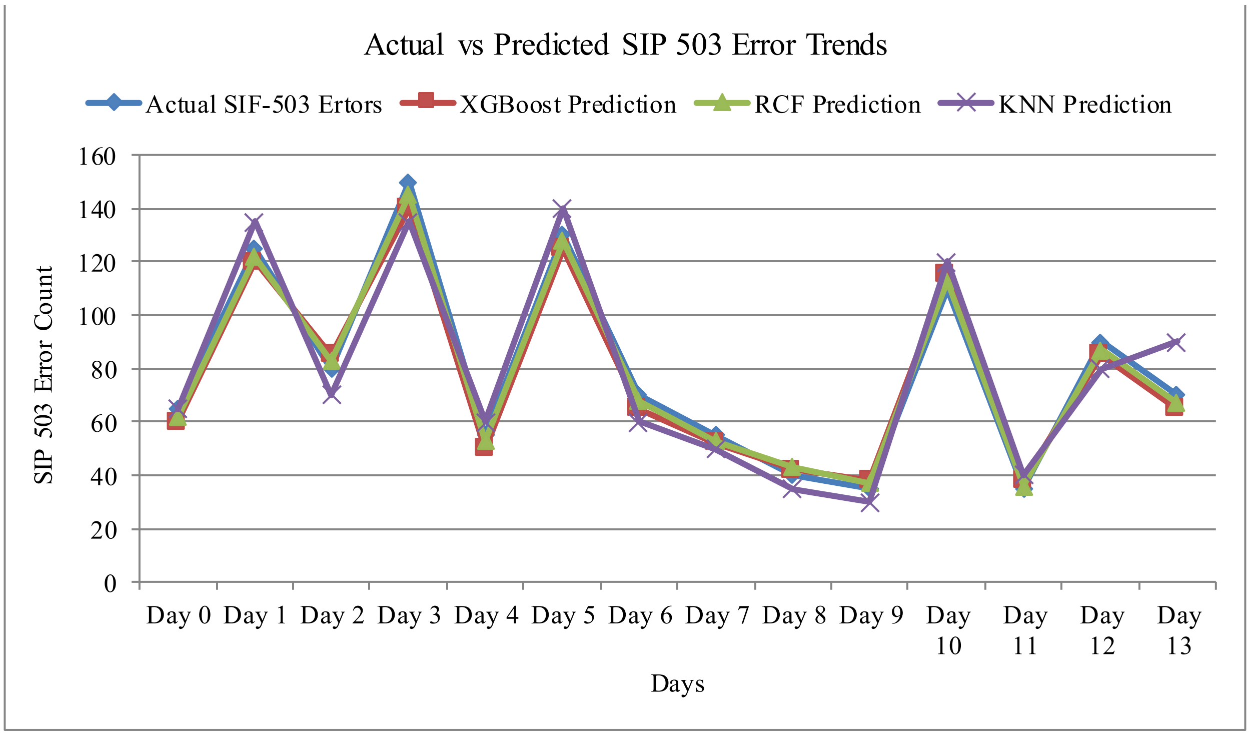

Fig. 6 SIP Error 503 pattern analysis and prediction by ML Models

The results indicate that the XG Boost model consistently provides superior predictive accuracy for SIP error codes 480 and 503, as demonstrated by higher R² scores and lower error metrics (MSE and RMSE) compared to Random Forest and KNN models. These findings highlight XG Boost's effectiveness in accurately forecasting and identifying SIP-related network anomalies, thereby offering robust support for proactive network management and improved user experience.

5.1.3. Case 3: Application congestion

To assess how well the models could detect congestion, a controlled experiment was carried out in which call traffic was steadily increased—from 500 to over 2,000 calls per hour. The test continued until the system reached its resource limits and began showing signs of overload. During this process, the system continuously logged critical metrics such as CPU usage, memory consumption, response times, and error rates.

These metrics were then analyzed using various machine learning models to pinpoint the exact threshold where application performance began to degrade. The results demonstrated how effectively each model could anticipate congestion and offer insights for proactive scaling. This kind of early detection is valuable for avoiding service interruptions and optimizing resource use before performance bottlenecks occur.

Table 3. CPU Overload monitoring and prediction via ML Models (Call processing)

Model	R^2 Score	MSE	RMSE
XGBoost	0.9289241126340176	3.7178398303332414	1.928170072979363
RCF	0.35117852299081787	33.938574942855944	5.825682358561608
KNN	0.031947429063369115	50.636925397110836	7.115962717518329

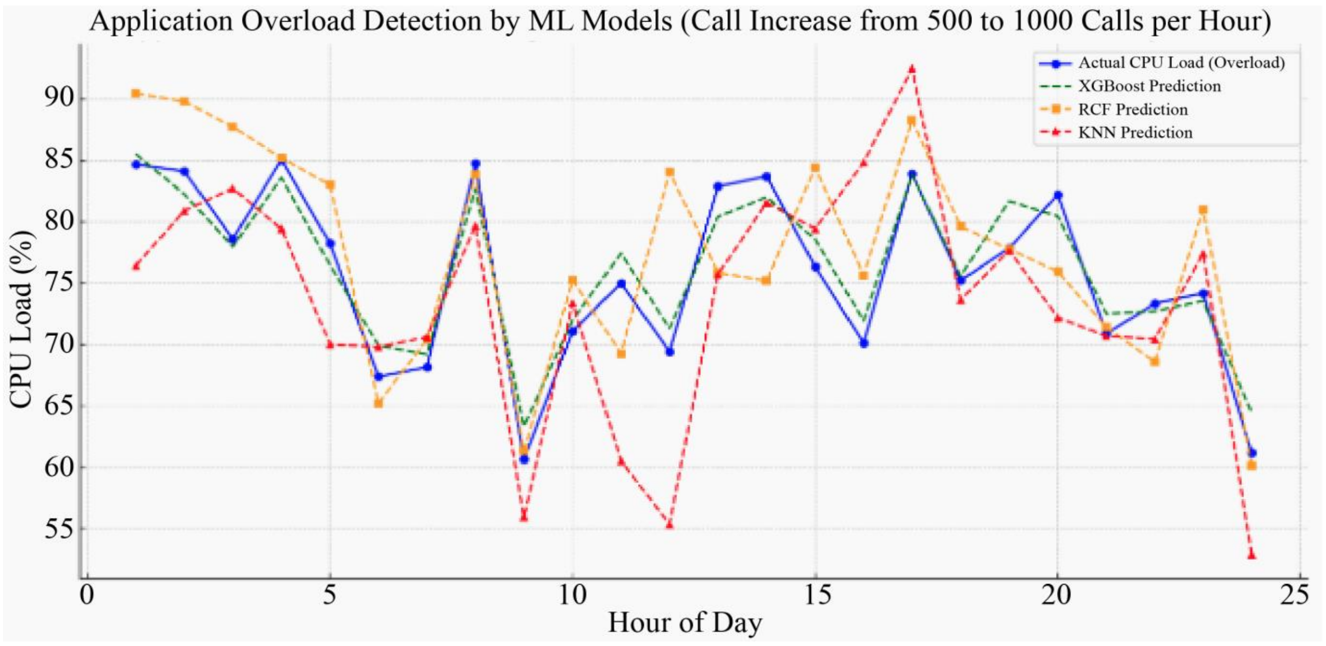

Fig. 7 CPU overload detection via ML models

Throughout this process, several critical performance indicators were continuously monitored, including CPU usage, memory consumption, latency in system response, and the frequency of application-level errors.

The collected data was then fed into the trained machine learning models to assess their ability to pinpoint when the system began to experience performance degradation due to overload.

Each model's output revealed how accurately it could identify the tipping point at which system resources became saturated. These insights proved valuable for enabling proactive resource scaling and effective management of system performance. The results showed clear differences in how each algorithm handled overload detection.

For example, XGBoost stood out by delivering precise predictions with lower error margins, as indicated by its high R² score and reduced Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). In contrast, the KNN model struggled under dynamic load conditions. Its predictions were more prone to inaccuracy, especially as call volume surged. This is likely due to KNN's reliance on local data points, which can limit its adaptability to rapidly changing system conditions.

Random Cut Forest (RCF) performed moderately well, detecting anomalies effectively but not as precisely as XGBoost. Unlike KNN, RCF is better suited for identifying outliers and early deviations, which gives it an advantage in spotting emerging congestion before full resource exhaustion occurs.

Overall, the experiment confirmed that models like XGBoost are more robust in anticipating and responding to overload conditions, offering strong potential for real-time congestion management in IMS-based cloud environments.

5.2. Generative AI for Network Optimization

Once anomalies are detected within the network, the system leverages generative AI to provide actionable guidance on how to adjust configuration settings in real time. These AI-driven recommendations may include retuning internal timers, prioritize certain types of resources, or even rolling back recent configuration changes that may have introduced instability.

What makes this system particularly effective is its integration with CI/CD pipelines and network orchestrators, allowing these suggested adjustments to be carried out automatically—without manual intervention. For example, during periods of high traffic when CPU usage crosses critical thresholds (typically around 70–80%), the system initiates auto-scaling processes to allocate additional application resources. Conversely, when traffic subsides and CPU usage drops below 50%, the system scales resources back down to conserve costs.

This dynamic adjustment process significantly enhances operational efficiency. It ensures that telecom services remain responsive and stable, even under varying demand levels, while also keeping infrastructure costs under control. In essence, generative AI enables a shift from reactive troubleshooting to proactive and autonomous network management.

6. Conclusion and Future Work

This study highlights the strong potential of using AI and machine learning techniques to improve the performance and scalability of IMS applications running on cloud platforms. The models demonstrated reliable predictive capabilities, helping telecom systems respond more intelligently to network conditions and resource demands.

Future research could focus on refining these predictive models through closer integration with live, real-time data streams. Enhancing the role of generative AI will also be key, especially for enabling more autonomous decision-making in network configuration and management.

A unified analytics toolset across network monitoring systems would bring added value—making it easier to correlate data from different parts of the infrastructure and apply machine learning models more consistently. This level of integration could lead to even more accurate forecasting and faster, smarter responses to changes in the network.

One particularly promising area for future development is the optimization of Radio Access Networks (RAN), where AI/ML techniques could help manage user mobility more effectively. By analyzing network behavior and predicting user state handovers in advance, these models could ensure smoother transitions between cells, ultimately improving service quality and user experience.

References

[1] Haris Haskić, and Amina Radončić, "The Effects of 5G Network on People and the Environment: A Machine Learning Approach to the Comprehensive Analysis," *World Journal of Advanced Engineering Technology and Sciences*, vol. 11, no. 1, pp. 301-309, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[2] Hongji Huang et al., "Deep Learning-Based Millimeter-Wave Massive MIMO for Hybrid Precoding," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 3027-3032, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[3] Ali Yadavar Nikravesh et al., "Mobile Network Traffic Prediction using MLP, MLPWD, and SVM," *IEEE International Congress on Big Data*, San Francisco, CA, USA, pp. 402-409, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[4] Pathak et al., "Quantum Models Enhancing Predictive Accuracy in 5G Resource Allocation," *Quantum Journal of Communications,* vol. 12, no. 3, pp. 210-218, 2024.

[5] Jun-Bo Wang et al., "A Machine Learning Framework for Resource Allocation Assisted by Cloud Computing," *IEEE Network*, vol. 32, no. 2, pp. 144-151, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[6] Noman Abid, "Enhanced IoT Network Security with Machine Learning Techniques for Anomaly Detection and Classification," *International Journal of Current Engineering and Technology*, vol. 13, no. 6, pp. 536-544, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Ramraj Dangi et al., "ML-Based 5G Network Slicing Security: A Comprehensive Survey," *Future Internet*, vol. 14, no. 4, pp. 1-28, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Sungmoon Kwon et al., "Towards 5G-Based IoT Security Analysis Against Vo5G Eavesdropping," *Computing*, vol. 103, pp. 425-447, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] Jiaying Yao et al., "A Robust Security Architecture for SDN-based 5G Networks," *Future Internet*, vol. 11, no. 4, pp. 1-14, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[10] Miranda McClellan, Cristina Cervelló-Pastor, and Sebastià Sallent, "Deep Learning at the Mobile Edge: Opportunities for 5G Networks," *Applied Sciences*, vol. 10, no. 14, pp. 1-27, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Hajiar Yuliana, Iskandar, and Hendrawan, "Comparative Analysis of Machine Learning Algorithms for 5G Coverage Prediction: Identification of Dominant Feature Parameters and Prediction Accuracy," *IEEE Access*, vol. 12, pp. 18939-18956, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Aishwarya Sandeep Desai, and Hrishikesh Mogare, "Machine Learning Approaches in 5G Networks," *International Journal for Research in Applied Science & Engineering* Technology, vol. 12, no. 6, pp. 1691-1697, 2024. [CrossRef] [Publisher Link]

[13] Hasna Fourati, Rihab Maaloul, and Lamia Chaari, "A Survey of 5G Network Systems: Challenges and Machine Learning Approaches," *International Journal of Machine Learning and Cybernetics*, vol. 12, pp. 385-431, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Param Pathak et al., "Resource Allocation Optimization in 5G Networks Using Variational Quantum Regressor," *International Conference on Quantum Communications, Networking, and Computing*, Kanazawa, Japan, pp. 101-105, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] Natalia Yarkina et al., "Performance Assessment of an ITU-T Compliant Machine Learning Enhancements for 5G RAN Network Slicing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 719-736, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Xu Gao, Jianfeng Wang, and Mingzheng Zhou, "The Research of Resource Allocation Method Based on GCN-LSTM in 5G Network," *IEEE Communications Letters*, vol. 27, no. 3, pp. 926-930, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] E. Selvamanju, and V. Baby Shalini, "Archimedes Optimization Algorithm with Deep Belief Network Based Mobile Network Traffic Prediction for 5G Cellular Networks," *4th International Conference on Smart Systems and Inventive Technology*, **Tirunelveli, India**, pp. 370-376, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] William M. Hayes, and Douglas H. Wedell, "Testing Models of Context-Dependent Outcome Encoding in Reinforcement Learning," *Cognition*, vol. 230, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Himanshu Sinha, "Analysis of Anomaly and Novelty Detection in Time Series Data Using Machine Learning Techniques," *Multidisciplinary Science Journal*, vol. 7, no. 6, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] V.N. Ganapathi Raju et al., "Study the Influence of Normalization/Transformation Process on the Accuracy of Supervised Classification," *Proceedings of the 3rd International Conference on Smart Systems and Inventive Technology*, Tirunelveli, India, pp. 729-735, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[21] Sahar Imtiaz et al., "Random Forests Resource Allocation for 5G Systems: Performance and Robustness Study," *IEEE Wireless Communications and Networking Conference Workshops*, Barcelona, Spain, pp. 326-331, 2018. [CrossRef] [Google Scholar] [Publisher Link]