Understanding the challenges of the call drop prediction problem in IP Multimedia Subsystem Networks

Pedro Victor Dos Santos Matias[1], **Ricardo Miranda Filho**[1], **Rosiane de Freitas**[1]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)

{pvsm, ricardo.filho, rosiane}@icomp.ufam.edu.br

Abstract. *Call drops in mobile networks using IMS (IP Multimedia Subsystem) technologies like Voice over LTE (VoLTE), Voice over New Radio (VoNR), and Voice over Wi-Fi (VoWiFi) present significant challenges to maintaining Quality of Service (QoS) and Quality of Experience (QoE). These failures often occur due to network congestion, weak signals, or issues related to software problems and complex situations. This study evaluates the effectiveness of machine learning models—Logistic Regression, Decision Tree, and XGBoost—in predicting call drops using a large dataset from Android devices, which had an imbalanced distribution of data. XGBoost achieved the highest overall accuracy but struggled with detecting rare call drops due to data imbalance. Although resampling techniques improved the detection of these call drops, they decreased overall accuracy, which remains a challenge. The proprietary nature of the dataset, which only provides information at the moment of disconnection, limits understanding of the entire call performance and the changes that occur during the call. Future work should focus on improving the data collection process and exploring deep learning techniques to capture complex patterns and improve prediction accuracy.*

1. Introduction

Modern telephony has moved from traditional circuit-switched networks to IP-based technologies like Voice over LTE (VoLTE), Voice over Wi-Fi (VoWiFi), and Voice over New Radio (VoNR), all supported by the IP Multimedia Subsystem (IMS). These technologies allow voice communication over data networks, which has improved connectivity but also brought new challenges in maintaining Quality of Service (QoS) and Quality of Experience (QoE) [Elbayoumy et al. 2018]. One of the main problems is call drops, where calls end unexpectedly (illustrated in Figure 1), which can happen due to things like network congestion, weak signals, or issues with resource allocation [G V and Kumari P. 2023, Erunkulu et al. 2019a].

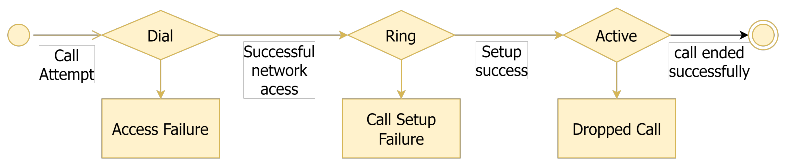

Figure 1. Generic Call Flow Process in Mobile Networks.

Using big data analytics to create predictive models has become an important tool to find patterns that can lead to call drops. These models help the network to be more reliable, allowing issues to be fixed before they affect the service [Kibria et al. 2018]. However, many studies only use data from specific network operators, which limits the models because they are based on information from just one source [Bahaa et al. 2022].

This work tries to address these gaps by using a proprietary dataset from a major manufacturer, which includes a variety of call events and parameters collected from Android devices. Because the dataset has a significant class imbalance, we used techniques like resampling and class weighting to help the models learn better. Our study not only looks at how well these techniques work but also creates a starting point for future research in predicting call drops using real data from users. Them, this research was guided by these main questions:

- How effective are machine learning models in predicting call drops based on disconnection call data from Android devices?
- Which preprocessing techniques work best to deal with class imbalance in this scenario?

To answer these questions, a pipeline was implemented to train four machine learning models: Logistic Regression, Decision Tree, Random Forest, and XGBoost (Extreme Gradient Boosting). The models were chosen because they are widely used in binary classification problems and serve as a baseline for future comparisons. To handle imbalanced data, we applied three techniques — undersampling, oversampling, and class weighting — resulting in a total of 12 experiments.

The remainder of this paper is organized as follows: Section 2 reviews related work in call drop prediction, including achievements and limitations. Section 3 presents the methodology used in our study. In Section 4, we discuss the results of the experiments, comparing the performance of different models and sampling techniques. Finally, Section 5 concludes the paper with a discussion of the implications of our findings and suggestions for future research.

2. Related Works

Previous research on predictive models for mobile network call failures has been extensive but faced challenges with limited failure data and imbalanced datasets. While some studies explored comprehensive solutions, others focused on specific scenarios like handover interruptions. A detailed discussion follows, emphasizing connections and gaps among these studies.

Different studies have centered on predicting call drops by directly modeling failure events in specific network environments. For instance, Bahaa et al. introduced a methodology that combines machine learning models with feature selection techniques to predict call failures within IMS networks [Bahaa et al. 2022]. They developed eight machine learning models using four different classifiers—Decision Tree, Naive Bayes, K-Nearest Neighbor (KNN), and Support Vector Machine (SVM)—paired with two feature selection methods (Filter and Wrapper). Their results showed that the SVM classifier, combined with the Wrapper feature selection method, achieved the highest prediction accuracy of 97.5%. While their approach succeeded in identifying the root causes of call failures and provided valuable insights into multi-factorial issues, it relied on data traces from a single mobile operator, limiting its applicability to broader, multi-operator environments and scenarios with more varied network conditions.

Similarly, the work by Mishra and Yadav focused on predicting call drops during user mobility, with an emphasis on handover scenarios [Mishra and Yadav 2020]. They used an Artificial Neural Network (ANN) model to minimize interruptions by analyzing parameters such as signal strength and subscriber speed, achieving a 95% accuracy. However, like Bahaa et al., their study was constrained by a small dataset, which raises concerns about the robustness of the model in varied real-world conditions.

In contrast, other studies have taken a more traditional approach by addressing call drop prediction in older network technologies, which, while insightful, limits their relevance to modern systems. Erunkulu et al. applied an ANN to predict call drops in GSM networks, achieving 87.5% accuracy [Erunkulu et al. 2019b]. Although their research demonstrated the effectiveness of ANNs, the focus on outdated GSM technology makes it less applicable to current networks like VoLTE and 5G.

Expanding beyond traditional models, recent work has explored advanced machine learning techniques. Ashok introduced a deep learning model that integrates auto imputation with Bayes optimization and transfer learning, specifically designed to handle the large and complex datasets typical in modern networks [Ashok 2024]. The inclusion of the Hybrid Skill-Levy Search Algorithm further enhanced the model's efficiency. Despite these innovations, the model's complexity presents challenges for real-time deployment, and its practical effectiveness remains to be validated with real-world data.

In another approach focusing on specific network conditions, Holmbacka and Al-Thaedan et al. both explored the prediction of network performance metrics that indirectly relate to call drops [Holmbacka 2018, Al-Thaedan et al. 2023]. Holmbacka concentrated on predicting network alarms in LTE networks, which could signal potential failures leading to call drops. However, the study's limitation to LTE environments diminishes its applicability to newer technologies. Similarly, Al-Thaedan's work on downlink throughput prediction in 4G-LTE networks suggested a method for preventing call drops due to inadequate throughput, but it did not address call drop prediction directly, thus missing other critical factors contributing to call interruptions.

Additionally, others researches explored more specialized scenarios. Daróczy et al. incorporated Self-Organizing Networks (SON) aspects into their model for session drop prediction in LTE networks, enhancing its ability to predict drops in dynamic environments [Daróczy et al. 2015]. However, this innovation was not thoroughly validated in real-world scenarios, and its relevance to more modern network technologies remains untested. Qu's study, on the other hand, focused on deep learning-based prediction of communication quality in LTE-R (Railway) networks, which could influence call drop predictions in similar environments [Qu 2020]. Yet, the specificity of LTE-R networks limits the broader applicability of the findings.

Lastly, Mudaliyar et al. investigated machine learning models for the automatic healing of call drops in 5G networks [Mudaliyar et al. 2020]. This study represents a forward-looking approach to predictive maintenance and real-time network adjustments, although it lacks a detailed analysis of model accuracy and the challenges of deploying these models in live networks.

A recurring limitation in these studies is their reliance on datasets that are often restricted to specific operators or network conditions. This narrow focus makes it difficult for the models to generalize effectively across different network scenarios, limiting their use in real-world applications. Furthermore, while deep learning models offer potential for high accuracy, they often lack interpretability and are challenging to deploy in mobile environments with limited resources.

3. Methodology

This section outlines the methodologies employed in this research, including data description, preprocessing steps, techniques for handling imbalanced data, the machine learning models used, the experimental protocol, and the evaluation methods.

3.1. Data Description

The dataset used in this study consists of 100,477,198 entries with 21 features, including numerical, categorical, and boolean data types. The primary target variable is `is_drop`, a boolean indicator representing whether a call was dropped. The features include radio signal parameters (e.g., `rssi`, `rsrp`, `rsrq`), device-related features (e.g., `android version`, `build_code`), network-related features (e.g., `mcc`, `mnc`, `initialRAT`), and temporal features (e.g., `day_of_week`, `hour_of_day`). Table 1 provides a detailed description of the dataset. Due to the proprietary nature of the dataset, there are limitations in terms of full disclosure of specific details, which restricts the possibility of direct comparisons with other studies.

Table 1. Dataset Description.

Column Name	Data Type	Description
`android_version`	int8	Android OS version
`build_code`	int64	Unique identifier for the build version
`product_code`	int64	Unique product identifier
`initialRAT`	int8	Initial radio access technology used
`activeRAT`	int8	Active radio access technology
`disconnectRAT`	int8	Radio access technology at disconnection
`channel`	int16	Absolute Radio-Frequency Channel Number (ARFCN)
`band`	int16	Frequency band used
`rat_handover`	boolean	Indicator of a RAT handover
`mcc`	int32	Mobile country code
`mnc`	int32	Mobile network code
`day_of_week`	int8	Day of the week when the event occurred
`hour_of_day`	int8	Hour of the day when the event occurred
`wifi_st`	boolean	Indicator of Wi-Fi state
`ims_reg`	boolean	Indicator of IMS registration
`roam`	boolean	Indicator of roaming status
`duration_sec`	int32	Duration of the event in seconds
`rssi`	int16	Received signal strength indicator
`rsrp`	int16	Reference signal received power
`rsrq`	int16	Reference signal received quality
`is_drop`	boolean	Indicator of call drop

3.2. Data Preparation and Preprocessing

Initial data preparation included restricting anonymized recorded events, collected via instrumented code in Android software from version 12 to 14 and recorded in 2024, that were disconnected in one of the following technologies: LTE, LTE-CA (Carrier Aggegation), WiFi Calling and NR - handling missing values, feature engineering and splitting the dataset. Missing values in numeric and categorical features were dealt with by removing incomplete records, resulting in a final dataset with all the features needed for the models. The `plmn` resource was split into `mcc` and `mnc`, and duplicate records were removed.

To prepare the data for machine learning models, the dataset was divided into training and testing sets in a 70-30 ratio. This split was stratified based on the is_drop target variable to maintain a balanced distribution of call failures and successful calls in both sets. Stratification is essential in cases of class imbalance, as it helps prevent the model from becoming biased towards the majority class and improves its ability to accurately predict call failures.

3.3. Techniques for Handling Imbalanced Data

Given the significant imbalance in the dataset, particularly the low occurrence rate of call drops, 3 techniques were employed to mitigate this issue. Random Undersampling was used to reduce the number of instances from the majority class (successful calls), thereby balancing the dataset. This method helps to equalize class distribution but can result in the loss of potentially valuable information, which may affect the model's performance [Drummond and Holte 2005]. Conversely, Random oversampling was applied to increase the representation of the minority class (call drops) by duplicating existing instances. While this approach enhances balance, it also risks overfitting by repeating data without introducing new information [Chawla et al. 2002].

To further address the class imbalance, class weighting was implemented across all models. By assigning higher weights to the minority class, the models were incentivized to focus more on correctly predicting call drops, improving their performance on these rare but critical events [Kumar et al. 2022]. This technique is particularly effective in scenarios like ours, where the skewed class distribution can lead to a model that is biased towards predicting the majority class. By adjusting the importance of each class, class weighting helps to mitigate this bias while preserving the original data distribution.

While data balancing techniques can facilitate better model training by making the classes more representative, they may also distort the natural occurrence rates in the data. To prevent this distortion from affecting the final model evaluation, the test data used to assess the models remains unbalanced, reflecting the real-world distribution of call drops [Krawczyk 2016].

The combination of class balancing techniques, as proposed by He et al. (2008), has proven effective in various studies. The use of class weighting, together with oversampling techniques, mitigates the effects of class imbalance and improves the performance of classification models, particularly in terms of the sensitivity (recall) of the minority class [He and Garcia 2009].

3.4. Classifiers Used

In this study, we selected Logistic Regression, Decision Tree, Random Forest, and XGBoost as our baseline models. These algorithms were chosen for their ability to provide interpretable results and clear insights into the importance of parameters, which is essential when analyzing call drop scenarios using our dataset. The data consists of parameters collected from Android devices at the moment of call disconnection. Based on the approach discussed in Bahaa et al. (2022), we focused on these models to better understand the relationships within this specific dataset before exploring more complex machine learning techniques. This strategy allows us to establish a foundation for future enhancements and analysis of call drop patterns.

The parameters for the classifiers used were initially set empirically, with the intention of refining them through future fine-tuning. We selected average values within the recommended ranges provided by each model's documentation as a starting point. Additionally, the settings were adjusted based on the large number of instances to be processed during training. Parameters such as the choice of loss function and the number of iterations were configured through a series of preliminary experiments to ensure computational efficiency. Fine-tuning these parameters remains as planned work for future studies to further optimize the models' performance.

Logistic Regression, a well-established method for binary classification in linear problems, was one of the models used in this study. The model was configured with the L-BFGS optimizer, known for its efficiency in handling large datasets, and balanced class weights to address the issue of class imbalance. We selected the `lbfgs` solver due to its optimization algorithm, which approximates the BFGS method but with reduced memory usage, making it suitable for problems with many features [Morales and Nocedal 2011]. The model was also set with a maximum iteration limit of 1000 and a tolerance of 10^{-4}. To ensure reproducibility, a fixed random seed was applied.

The Decision Tree model, a fundamental algorithm in machine learning, was also evaluated. This model splits data into branches based on feature values, creating a tree-like structure where each internal node represents a decision and each leaf node represents the outcome. Decision Trees are highly interpretable and capable of handling both categorical and numerical data [Gilpin et al. 2018]. However, to prevent overfitting, which is a common issue when trees grow too deep [Bertsimas and Dunn 2017], the maximum tree depth was limited to 15. The model was further configured with balanced class weights and a fixed random seed for reproducibility.

Random Forest, an ensemble method, was another model evaluated in this study. It constructs multiple decision trees using random subsets of features and bootstrapped data [Lee et al. 2020], with the final prediction made by averaging or majority voting across trees. This approach reduces overfitting and improves accuracy [Lebedev et al. 2014]. Our Random Forest model was configured with a maximum tree depth of 15, 150 estimators, balanced class weights, and a warm start for robustness. Additionally, parallel processing was enabled to improve efficiency, and a fixed random seed was set for reproducibility. This model's ability to capture complex interactions and nonlinear patterns makes it particularly suitable for large, complex datasets, offering significant advantages over traditional statistical methods that often assume linear relationships [Sucahyo et al. 2024].

Finally, XGBoost, an efficient implementation of gradient boosting, was included in the study. XGBoost builds decision trees sequentially, optimizing the learning process by minimizing a loss function [Chen and Guestrin 2016]. The model was configured for training with a maximum tree depth of 15, 150 estimators, and a learning rate of 0.05 to control complexity and prevent overfitting. The XGBoost also utilized `logloss` as the evaluation metric for binary classification. To handle the imbalanced dataset, the `scale_pos_weight` parameter was adjusted, ensuring better representation of the minority class [Jiang 2024]. Like Random Forest, XGBoost employed parallel processing using all available cores to enhance computational efficiency. Additionally, automatic label encoding was disabled due to the dataset was already encoded.

3.5. Evaluation Methods

The performance of the machine learning models was evaluated using metrics suitable for imbalanced datasets: weighted accuracy, precision, recall, F1-score, and ROC-AUC. These metrics offer different perspectives on the model's performance, especially when class distribution is uneven. The following Table 2 summarizes each metric with its corresponding equation.

Table 2. Summary of evaluation metrics and their equations.

Metric	Description	Equation
Weighted Accuracy	Overall accuracy, adjusted for class distribution.	$\sum_{i=1}^{n} \frac{N_i}{N} \times \frac{TP_i+TN_i}{TP_i+TN_i+FP_i+FN_i}$
Precision	Proportion of true positives among all positive predictions.	$= \frac{TP}{TP+FP}$
Recall	Proportion of true positives among all actual positives.	$\frac{TP}{TP+FN}$
F1-Score	Harmonic average of precision and recall.	$2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$
ROC-AUC	Area under the ROC curve, representing the trade-off between TPR and FPR.	$\int_0^1 TPR\, d(FPR)$, TPR $= \frac{TP}{TP+FN}$

3.6. Technological Setup

All experiments were conducted on a workstation with the following specifications: a 12th Gen Intel® Core$^{\text{TM}}$ i9-12900HX processor at 2.30 GHz, 64.0 GB of RAM, and a 64-bit Windows 11 operating system. The environment was managed using Anaconda with Python 3.11.9, and key libraries used included Scikit-Learn, Imbalanced-Learn, XGBoost, Pandas, and NumPy.

3.7. Experimental Protocol

Finally, the experimental protocol was designed to systematically train and evaluate the performance of various machine learning models on an imbalanced dataset. The workflow for this protocol is visually summarized in Figure 2. The following steps summarize the procedure:

1. **Dataset Preparation**: Call records collected from March 9 to June 15 were preprocessed to extract Mobile Country Code (MCC) and Mobile Network Code (MNC), handle missing values, and encode categorical variables. The dataset was split into training and test sets using a 70/30 ratio, with a stratified hold-out approach to ensure class distribution consistency in both sets, which is crucial for imbalanced datasets.
2. **Handling Imbalanced Data**: To address the significant class imbalance, we applied undersampling, oversampling, and a baseline approach without resampling, using only balanced class weights. These methods aimed to improve the detection of rare call drops while maintaining overall model performance.

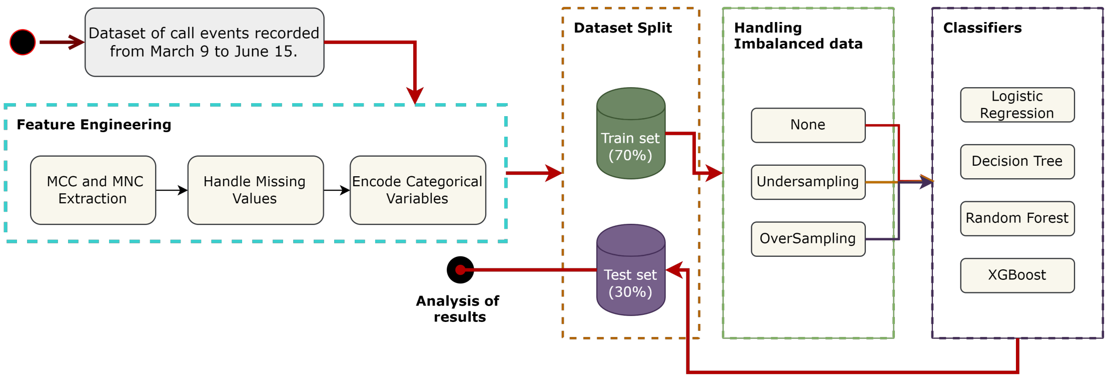

Figure 2. Workflow of the experimental protocol for training and evaluating machine learning models on the imbalanced dataset.

3. **Model Training and Evaluation**: The four classifiers were trained and evaluated using metrics like accuracy, precision, recall, F1-score, confusion matrices, and ROC-AUC curves. Although techniques like cross-validation or multiple hold-out iterations are recommended to reduce result variability, these were impractical due to the dataset's size and the long training times required. Instead, a single stratified hold-out split was used to balance computational feasibility with reliable performance evaluation.
4. **Exporting models**: The trained models were serialized using Pickle for storage and future use, allowing for integration into mobile applications for real-time call drop predictions. This ensures the models are reusable and supports ongoing validation and optimization.

4. Results and Discussion

After the training phase, the models were evaluated using a 30% holdout portion of the dataset. This section summarize the key findings from this evaluation, including insights from confusion matrices, area under the ROC curve (AUC), and the impact of resampling techniques.

XGBoost consistently outperformed the other models, as evidenced by its top performance in Table 3. The version Without resampling achieved the highest accuracy, demonstrating its effectiveness in this 'overall' classification task. However, while XGBoost excels overall, it struggled to identify a significant portion of `Drop` cases, as shown in the confusion matrix (Figure 3). This suggests that while it's highly accurate, its sensitivity(recall at 0.56) to detecting the minority class should be improved.

Random Forest exhibited a well-balanced performance, consistently achieving strong precision and recall metrics across both classes. Unlike XGBoost, Random Forest managed to achieve a higher recall for the `Drop` class, while still maintaining a moderate precision, similar to the XGboost's version with some sampling, that make Random Forest as another option when a more balanced approach is desired.

As expected, Logistic Regression underperformed due to its limitations in handling non-linear relationships and its sensitivity to outliers and irrelevant variables. Its inability to capture the intricate patterns present in the data, combined with its assump-

Table 3. Performance Metrics for Each Model and Sampling Technique.

Model	Resampling Method	Precision (Drop)	Weighted Precision	Recall (Drop)	Weighted Recall	F1-Score (Drop)	Weighted F1-score	Accuracy	ROC-AUC
Logistic Regression	No Resampling	0.20	0.73	0.49	0.56	0.28	0.61	0.561686	0.523882
Logistic Regression	Oversampling	0.20	0.73	0.49	0.56	0.28	0.61	0.561686	0.524122
Logistic Regression	Undersampling	0.20	0.73	0.49	0.56	0.28	0.61	0.561691	0.523940
Decision Tree	No Resampling	0.51	0.86	0.74	0.83	0.60	0.84	0.828342	0.878905
Decision Tree	Oversampling	0.51	0.86	0.74	0.83	0.60	0.84	0.829111	0.878826
Decision Tree	Undersampling	0.50	0.86	0.74	0.83	0.60	0.84	0.826618	0.878738
Random Forest	No Resampling	0.56	0.87	0.71	0.85	0.63	0.86	0.852428	0.885802
Random Forest	Oversampling	0.56	0.87	0.71	0.85	0.63	0.86	0.852652	0.885989
Random Forest	Undersampling	0.56	0.87	0.71	0.85	0.63	0.86	0.852806	0.885987
XGBoost	No Resampling	0.89	0.91	0.56	0.91	0.69	0.90	0.911098	0.913268
XGBoost	Oversampling	0.57	0.88	0.78	0.86	0.66	0.87	0.859735	0.913025
XGBoost	Undersampling	0.57	0.88	0.78	0.86	0.66	0.87	0.858139	0.912905

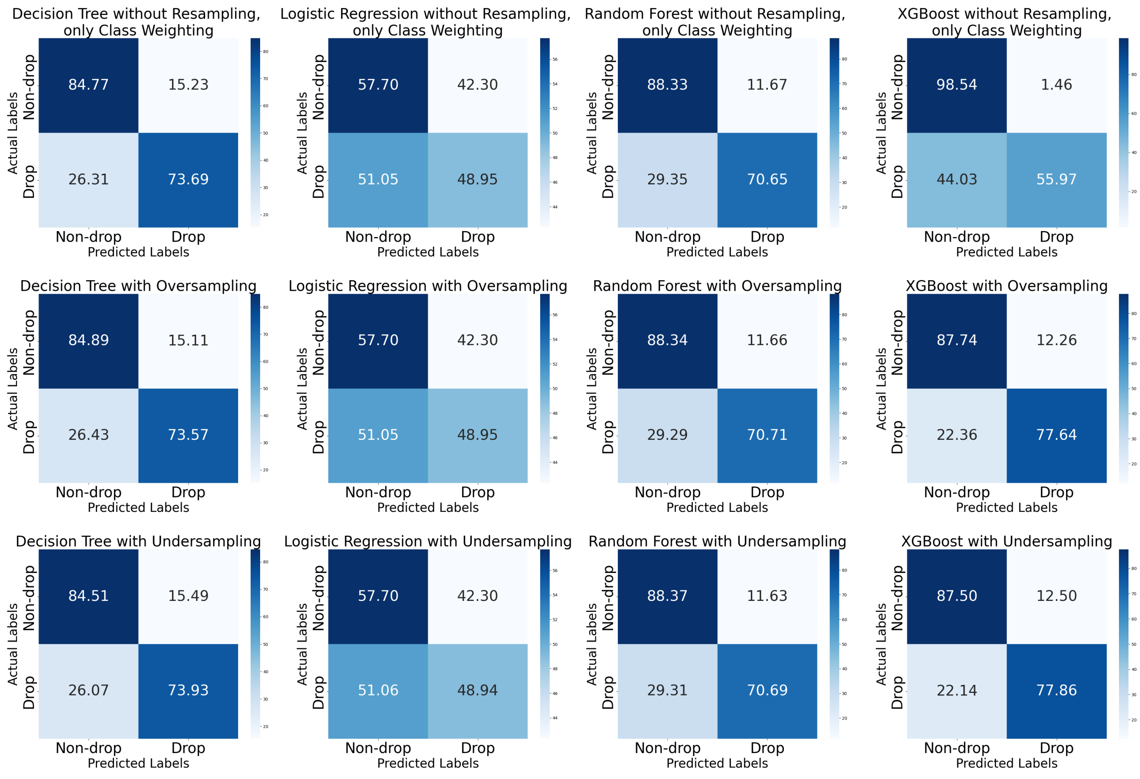

Figure 3. Confusion Matrices for Each Model and Sampling Technique.

tion of a linear decision boundary and independence between observations, made it less suited for the complexities of this dataset compared to more advanced models like XGBoost and Random Forest, based on their results.

Across all models, precision was consistently higher only for the "Non-drop" class, reflecting its greater accuracy in predicting instances of non-drop. On the other hand, revocation was generally slightly higher for the `Drop` class in the tree-based models, indicating their greater ability to identify the majority of actual “Drop” cases, despite some trade-offs in precision.

Regarding the impact of sampling, oversampling and undersampling had a minimal effect on the overall performance of the models, likely due to the large dataset size and the use of class weights. While XGBoost and Random Forest showed slight improvements in identifying `Drop` cases with resampling, these gains were not substantial enough to significantly alter the overall metrics. Given these results, focusing on fine-tuning the models to enhance their sensitivity to the minority class, rather than relying on resampling techniques, might be more effective. Adjustments such as optimizing class weights, lowering the decision threshold, or tweaking hyperparameters could potentially yield better recall for the `Drop` class without compromising the overall performance.

5. Concluding Remarks

This study aimed to understand the challenges of predicting call drops in IP Multimedia Subsystem (IMS) networks, focusing on the inherent complexities of this problem using a large and complex dataset obtained from a significant number of Android devices. The study evaluated four machine learning models—Logistic Regression, Decision Tree, Random Forest, and XGBoost—while addressing class imbalance through undersampling, oversampling, and class weighting. These challenges are partly due to the intrinsic difficulties in collecting and categorizing data, as well as the non-obvious causes that may underlie some call drops, which were not captured by the models during training, possibly due to a lack of sufficient examples or the inherent limitations of these less complex models.

One of the research questions was to determine which model performed best in predicting call drops, especially given the challenges posed by unbalanced data. Among the models tested, XGBoost emerged as the most accurate, achieving the highest ROC-AUC scores, particularly in scenarios without resampling. However, despite its strong overall performance, XGBoost had difficulty correctly identifying the less frequent call drops, suggesting a limitation in its sensitivity to minority classes. Random Forest provided a more balanced performance between accuracy and recall, making it a more dependable choice when both metrics are important. Logistic Regression, as expected, underperformed, likely because it struggled to capture the complex patterns present in the data. Although resampling techniques offered some improvements, they were insufficient to significantly enhance the models’ ability to detect the rare call drop events, highlighting the ongoing challenge in addressing data imbalance.

The proprietary nature of the dataset limits the understanding of the entire call behavior, as it only captures data at the endpoint (end user) at the time of call disconnection, without providing information on the complete lifecycle of the call. This gap prevents a full understanding of the issues that can arise throughout the call, including factors related to operator metrics and resource allocation. A more detailed dataset would be essential to fully analyze the underlying causes of call drops. Additionally, the study highlights the need for more advanced modeling techniques. While traditional machine learning models provide a baseline, exploring deep learning approaches, such as Multi-Layer Perceptrons (MLPs) or other techniques, could better capture the complex patterns in the data.

Future efforts should refine these models to balance accuracy and recall, reduce false positives, and improve data collection with better collectors to capture the full call lifecycle. Addressing these challenges is key to improving the detection of call drops, especially those with non-obvious causes.

Acknowledgements

This research, part of the ”SWPERFI - Artificial Intelligence Techniques for Analysis and Optimization of Software Performance” project, was supported by Motorola Mobility Comércio de Produtos Eletrônicos Ltda and Flextronics da Amazônia Ltda under Agreement No. 004/2021 with ICOMP/UFAM, in accordance with Federal Law No. 8.387/1991. Additional support was provided by Brazilian agencies CAPES (Finance Code 001), CNPq, and FAPEAM through the POSGRAD project 2024/2025.

References

Al-Thaedan, S. et al. (2023). Downlink throughput prediction using machine learning models on 4g-lte networks. *IEEE Access*, 11:32345–32356.

Ashok, K. (2024). A deep auto imputation integrated bayes optimized transfer learning model with hybrid skill-levy search algorithm (dai-bots) for call drop prediction in mobile networks. *Journal of Communication and Information Systems*, 39:120–130.

Bahaa, A., Shehata, M., Gasser, S. M., and El-Mahallawy, M. S. (2022). Call failure prediction in ip multimedia subsystem (ims) networks. *Applied Sciences*, 12(16):8378.

Bertsimas, D. and Dunn, J. (2017). Optimal classification trees. *Machine Learning*, 106:1039–1082.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Daróczy, T. et al. (2015). Machine learning based session drop prediction in lte networks and its son aspects. *IEEE Communications Letters*, 19(5):822–825.

Drummond, C. and Holte, R. C. (2005). Severe class imbalance: Why better algorithms aren’t the answer. In *European Conference on Machine Learning*, pages 539–546. Springer.

Elbayoumy, A. D., Hussein, M., and Al-Ashry, S. F. (2018). Ott voip over lte vs. volte end-to-end qos using opnet. In *The International Conference on Electrical Engineering*, volume 11, pages 1–14. Military Technical College.

Erunkulu, O. O., Onwuka, E. N., Ugweje, O. C., and Ajao, L. A. (2019a). Prediction of call drops in gsm network using artificial neural network. *Jurnal Teknologi Dan Sistem Komputer*, 7:38–46.

Erunkulu, T. A. et al. (2019b). Prediction of call drops in gsm network using artificial neural network. *International Journal of Mobile Network Design and Innovation*, 10(3):150–160.

G V, A. and Kumari P., V. (2023). A novel chimp optimized linear kernel regression (colkr) model for call drop prediction in mobile networks. *International Journal on Recent and Innovation Trends in Computing and Communication*, 11:593–603.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M. A., and Kagal, L. (2018). Explaining explanations: an overview of interpretability of machine learning. *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA).*

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

Holmbacka, S. (2018). Alarm prediction in lte networks. *International Journal of Mobile Network Design and Innovation*, 12(4):345–356.

Jiang, Y. (2024). Predicting loan default: a comparative analysis of multiple machine learning models. *Highlights in Science, Engineering and Technology*, 85:169–175.

Kibria, M. G., Nguyen, K., Villardi, G. P., Zhao, O., Ishizu, K., and Kojima, F. (2018). Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks. *IEEE Access*, 6:32328–32338.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in artificial intelligence*, 5(4):221–232.

Kumar, V., Lalotra, G. S., Sasikala, P., Rajput, D. S., Kaluri, R., Lakshmanna, K., Shorfuzzaman, M., Alsufyani, A., and Uddin, M. (2022). Addressing binary classification over class imbalanced clinical datasets using computationally intelligent techniques. In *Healthcare*, volume 10, page 1293. MDPI.

Lebedev, A., Westman, E., Van Westen, G., Kramberger, M., Lundervold, A., Aarsland, D., Soininen, H., Kłoszewska, I., Mecocci, P., Tsolaki, M., et al. (2014). Random forest ensembles for detection and prediction of alzheimer's disease with a good between-cohort robustness. *NeuroImage: Clinical*, 6:115–125.

Lee, T.-H., Ullah, A., and Wang, R. (2020). Bootstrap aggregating and random forest. *Macroeconomic forecasting in the era of big data: Theory and practice*, pages 389–429.

Mishra, S. and Yadav, P. (2020). Mobility robustness optimization using ann for call drop prediction. *IEEE Transactions on Vehicular Technology*, 69(8):8345–8354.

Morales, J. L. and Nocedal, J. (2011). Remark on "algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization". *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–4.

Mudaliyar, R. et al. (2020). Machine learning based call drop healing in 5g. *Journal of Communication and Information Systems*, 35:145–155.

Qu, J. (2020). Temporal-spatial collaborative prediction for lte-r communication quality based on deep learning. *Wireless Networks*, 26:1925–1936.

Sucahyo, C. B., Rizqini, F. Q., Naufal, A., Yandratama, H., Shiddiqy, J. A., Utama, A. B. P., Putri, N. S. F., and Wibawa, A. P. (2024). Performance analysis of random forest on quartile classification journal. *Applied Engineering and Technology*, 3(1):1–15.