NOVA IMS Information Management School

MDSAA

Master Degree Program in

Data Science and Advanced Analytics

From Reactive to Proactive

A Journey towards Issue Resolution with Explainable Neural Networks in the Healing Project

Filipe Duarte Pereira

Project Work

presented as partial requirement for obtaining a Master's Degree in Data Science and Advanced Analytics

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

From Reactive to Proactive

A Journey towards Issue Resolution with Explainable Neural Networks in the Healing Project

by

Filipe Duarte Pereira

Project Work presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics, with a specialization in Data Science

Supervised by

Fernando José Ferreira Lucas Bação, PhD, NOVA Information Management School

April, 2025

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Lisbon, April 2025

ABSTRACT

In the increasingly competitive telecommunications sector, ensuring customer satisfaction hinges on the ability to detect and resolve technical issues before they impact the user experience. Traditional reactive strategies often fail to address the needs of silent customers, those who do not report service disruptions yet contribute to hidden churn. This thesis investigates the use of explainable neural networks to shift from reactive to proactive issue resolution within the context of the Healing Project at NOS, a leading telecommunications provider in Portugal. Leveraging tabular data derived from network telemetry and customer interactions, this research explores the application of intrinsically interpretable and post-hoc explainability techniques to neural network models. The study applies a CRISP-DM methodology to structure the data science process, from business understanding to model evaluation. The performance of the explainable neural networks is benchmarked against state-of-the-art models, with a focus on balancing predictive accuracy and transparency. A human-centered evaluation involving domain experts further assesses the usability and trustworthiness of model explanations. Results demonstrate that certain explainable neural networks can achieve predictive performance comparable to state-of-the-art models while offering actionable insights that support decision-making. This work contributes to the broader adoption of Explainable AI (XAI) in telecommunications by showcasing the feasibility of interpretable neural networks for proactive customer support. It also highlights the importance of evaluating interpretability in real-world operational settings, where transparency and accountability are critical to building trust in automated systems.

KEYWORDS

Explainable Neural Networks; Silent Customer Churn; Proactive Issue Resolution; Telecommunications; Interpretable Machine Learning; Human-Centered AI

Sustainable Development Goals (SDG):

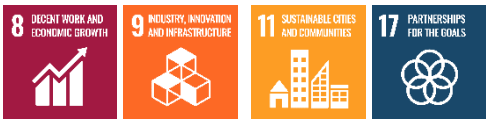

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI Artificial Intelligence – The simulation of human intelligence by machines, especially computer systems.

CRISP-DM Cross-Industry Standard Process for Data Mining – A methodology used to structure data science projects.

IG Integrated Gradients – A post-hoc method for attributing the prediction of a neural network to its input features.

IMN Interpretable Mesomorphic Networks – A type of neural network designed for interpretability on tabular data.

NBM Neural Basis Models – A class of interpretable neural networks that utilize basis functions for explanation.

OT Ordem de Trabalho (Work Order) – A record in the customer service system indicating a scheduled intervention or service issue.

PT Participação Técnica (Technical Report) – A customer-initiated complaint or service request involving technical issues.

XAI Explainable Artificial Intelligence – A set of methods and practices for making AI decisions interpretable by humans.

1. INTRODUCTION

In today's fast-paced telecommunications landscape, customer satisfaction remains essential for sustained success. The ability to identify and address technical issues proactively is becoming increasingly critical as traditional reactive approaches are no longer enough to meet customer expectations. NOS, a leading telecommunications company in Portugal, faces the challenge of transforming its issue resolution practices to strengthen customer trust and retention. This thesis explores the potential of leveraging explainable neural networks to predict and resolve technical issues before they impact customer experience, marking a significant departure from conventional methodologies.

1.1. BACKGROUND

In the telecommunications industry, customer satisfaction is a critical driver of business success (Khatibi et al., 2002). However, maintaining high satisfaction levels has become increasingly complex due to rapid technological advancements and rising customer expectations (Amin et al., 2018). Even minor technical issues can lead to significant dissatisfaction, often driving customers to seek alternatives in a highly competitive market (Roy & Ganguli, 2008). One of the most pressing consequences of dissatisfaction is customer churn, the rate at which customers discontinue their service and switch to competitors. Churn is particularly problematic because acquiring new customers is far more expensive than retaining existing ones (Kim & Yoon, 2004; Singh, 2021).

A key challenge in mitigating churn lies in addressing the needs of silent customers who experience technical issues but do not report them (Al_Janabi & Razaq, 2019). Instead of seeking help, they quietly churn, leaving service providers with no opportunity to resolve their issues. Silent customers represent a hidden yet significant risk, as their dissatisfaction remains undetected until it translates into revenue loss (Fornell, 1992; Reichheld & Sasser, 1990)

This reality underscores the importance of transitioning from reactive issue resolution, which addresses problems only after they arise (Cherukuri et al., 2020), to proactive strategies that anticipate and mitigate issues before they affect the customer experience (Ochuba et al., 2024). Traditional methods of problem detection, often reliant on post-incident data analysis, fail to address silent customer needs or prevent disruptions that erode customer trust (Al-Mashraie et al., 2020; Ascarza et al., 2018). Emerging technologies in data science, particularly machine learning and neural networks, offer powerful tools for preemptive problem detection and resolution. These technologies analyze vast amounts of data to uncover subtle patterns indicative of dissatisfaction or technical issues, enabling service providers to act before problems escalate (Barkai & Harison, 2011).

However, applying these technologies in real-world contexts such as telecommunications comes with challenges. Simpler machine learning models generally provide better interpretability, allowing for easier understanding of their predictions. Yet, these models often struggle to achieve the performance required for complex and nonlinear problems. Neural networks, on the other hand, excel in identifying complex patterns and delivering higher predictive accuracy but are frequently criticized for being "black boxes" (Carvalho et al., 2019). Their predictions are difficult to interpret, limiting their practical value for business stakeholders who require transparent and actionable outcomes (Linardatos et al., 2021).

To address these limitations, Explainable Artificial Intelligence (XAI) has emerged as a critical field, focusing on making machine learning and neural network models more interpretable and their results more understandable (Rai, 2020). XAI bridges the gap between performance and transparency, enabling technical teams to leverage the predictive power of neural networks while providing business teams with insights they can trust and act upon (Shah & Konda, 2021). Despite advancements in XAI methodologies, their application in proactive issue resolution, especially using explainable neural networks, remains underexplored in the telecommunications sector. This gap is particularly pronounced when it comes to addressing silent customer churn, where timely, interpretable insights are essential to enable effective intervention.

This study aims to address this research gap by focusing on the proactive identification and resolution of technical issues using explainable neural networks. By applying advanced anomaly detection methods to identify patterns indicative of technical problems and leveraging explainability methods to interpret model outputs, this research seeks to contribute to the broader adoption of XAI in telecommunications. Ultimately, it aims to improve customer satisfaction and retention by enabling timely, data-driven interventions that resonate with both technical and business stakeholders, reducing churn and addressing the hidden challenges posed by silent customers.

1.2. Research Question and Objectives

This thesis explores the intersection of predictive modeling and model interpretability, with a focus on applying neural networks to tabular data derived from network telemetry and customer interactions. The central research question guiding this work is:

How can explainable neural network models be leveraged to proactively detect customer-perceived technical issues in telecommunications, and what is the trade-off between predictive accuracy and interpretability in these models when applied to tabular data?

This question captures two key challenges. The first is the need for high-performing models capable of identifying customer-experienced issues before they escalate. The second is the

need for explanations that are clear and trustworthy enough to support operational decision-making by business stakeholders.

To address these challenges, the research is guided by specific objectives. The first objective is to develop neural network models capable of detecting early signals of technical issues that are likely to be perceived and reported by customers. This involves applying anomaly detection and supervised learning methods tailored to structured, high-dimensional data. The second objective is to explore and evaluate different explainability techniques, both intrinsic and post-hoc, for neural networks applied to tabular data, with a focus on identifying methods that produce actionable and understandable insights without significantly compromising model performance. Finally, the third objective is to analyze the trade-off between predictive accuracy and interpretability, assessing the practical usability of explainable models in real-world telecommunications operations through both quantitative metrics and human-centered evaluations.

By addressing these objectives, this study aims to contribute both technically robust models and practically usable tools that help telecommunications providers proactively manage customer experience and reduce churn.

1.3. Defining the Scope

To ensure a focused and actionable investigation, the scope of this thesis is narrowed in several ways. First, the research is confined to the telecommunications sector, using NOS as a case study, ensuring that the findings are directly relevant to the industry's unique challenges. Second, the study emphasizes a balance between predictive accuracy and interpretability, prioritizing models that not only predict technical issues with high accuracy but also provide insights that are easily understood and trusted by business stakeholders. Third, the research centers on proactive resolution, focusing on identifying and addressing issues before they escalate, rather than engaging in reactive or post-incident analysis. By addressing these well-defined questions within the outlined scope, this thesis aims to make a meaningful contribution to both academic literature and practical applications, advancing the integration of explainable AI in the telecommunications sector.

1.4. Outline

The first chapter introduces the context and motivation for this research, highlighting the importance of customer satisfaction in the telecommunications industry and the need for proactive issue resolution. It presents the research question, objectives, and scope of the study, emphasizing the role of explainable neural networks in identifying customer-perceived technical issues.

The second chapter presents a focused literature review on the principles and methodologies of explainable artificial intelligence (XAI), with a particular emphasis on neural networks for tabular data. It explores the challenges of interpretability in deep learning models, categorizes existing explainability approaches and critically reviews state-of-the-art techniques used for explaining neural network decisions.

The third chapter describes the research methodology, structured according to the CRISP-DM framework. It outlines the business understanding and data context provided by NOS, followed by data exploration, labeling strategies, and feature selection. It then details the model development process, including benchmark models and selected explainable neural networks. Finally, it explains the evaluation strategy, combining quantitative metrics and human-centered interpretability assessment.

The fourth chapter presents and discusses the results of the developed models, comparing their performance in terms of accuracy, precision, recall, and interpretability. It highlights the trade-offs between model transparency and predictive power, and identifies the most effective approaches for operational use at NOS.

The fifth and final chapter concludes the thesis by summarizing the key findings and contributions of the work. It reflects on the practical implications for telecommunications providers, and proposes directions for future research, particularly in extending explainable AI techniques for broader use in proactive customer support systems.

2. LITERATURE REVIEW

This chapter reviews the existing literature on explainable neural networks, with a particular focus on applications involving tabular data. It first introduces the foundations of neural networks, their strengths, and the challenges they present in terms of interpretability. It then explores the dimensions and methods used for explainability in neural networks. Finally, it reviews state-of-the-art methods developed to enhance the transparency of neural networks for tabular data, highlighting existing research gaps that motivate this thesis.

2.1. EXPLAINABLE NEURAL NETWORKS

2.1.1. Neural Networks

Neural networks are a class of machine learning models designed to mimic the functioning of the human brain by learning patterns from data. They consist of layers of artificial neurons, which process information by applying weighted connections and activation functions. These models are particularly effective in capturing complex relationships, handling high-dimensional data, and making predictions based on learned patterns (Dongare et al., 2012).

A typical neural network consists of three key layers. The input layer receives the raw data and forwards it through the network. Hidden layers perform computations by extracting patterns and learning hierarchical representations of the data. Finally, the output layer generates the prediction. Neurons in each layer are connected to those in the next through weighted links, which are adjusted during training using optimization algorithms. Activation functions introduce non-linearity within each neuron, allowing the network to model complex patterns that linear models cannot capture (Dongare et al., 2012). An illustration of a simple neural network architecture is presented in Figure 2.1.

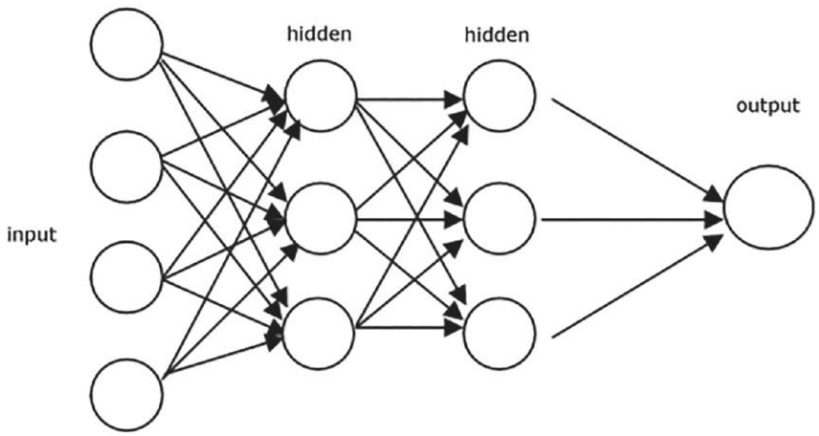

Figure 2.1 – Neural Network Architecture

According to Haykin (1994), neural networks have become increasingly popular due to their ability to process large volumes of data and uncover complex relationships that traditional models may overlook. Their main advantages include high expressive power, which enables them to learn non-linear patterns and feature interactions that make them superior to linear models in many applications. They are also highly scalable, performing well with large-scale datasets by leveraging advances in computational power and parallel processing. Another advantage is their ability to learn relevant features automatically from raw data, which reduces the need for manual feature engineering, particularly in high-dimensional spaces. Finally, neural networks offer strong generalization capabilities, allowing them to perform well on unseen data when properly trained and regularized.

Despite these strengths, neural networks are often criticized for being "black-box" models, meaning their internal decision-making processes are difficult to interpret. This lack of transparency can reduce trust, limit adoption in high-stakes domains, and pose challenges for regulatory compliance. Consequently, explainability techniques have become essential for making neural networks more interpretable, aiming to bridge the gap between model accuracy and trustworthiness (Angelov & Soares, 2020).

2.1.2. Dimensions of Explainability in Neural Networks

The challenge of neural network interpretability has led to the development of various dimensions of explainability, which can be categorized based on their approach and scope. One important distinction is between intrinsic (by-design) and post-hoc explainability. Intrinsic explainability refers to models that are inherently interpretable, meaning their structure is designed from the outset to be transparent. These models provide insights directly through their architecture, without requiring additional interpretability methods (Das & Rad, 2020). In contrast, post-hoc explainability involves applying techniques after a model has been trained, offering interpretations that aim to explain the model's decisions without altering its original structure (Das & Rad, 2020).

Another key categorization distinguishes between model-specific and model-agnostic methods. Model-specific explainability techniques are tailored to particular types of models, leveraging their internal mechanics to generate explanations that are often more faithful and efficient (Molnar, 2020). Model-agnostic methods, on the other hand, are designed to be applicable to any machine learning model. They do not depend on the internal workings of the model but instead approximate its decision boundaries externally (Molnar, 2020).

Finally, explainability methods can be divided into local and global approaches. Local explainability focuses on explaining individual predictions, answering specific questions such as "Why was this instance classified in this way?" (Das & Rad, 2020). Global explainability seeks to provide a broader understanding of how a model makes decisions across the entire dataset,

identifying the most influential features and uncovering general decision patterns (Das & Rad, 2020). These three dimensions are summarized visually in Figure 2.2.

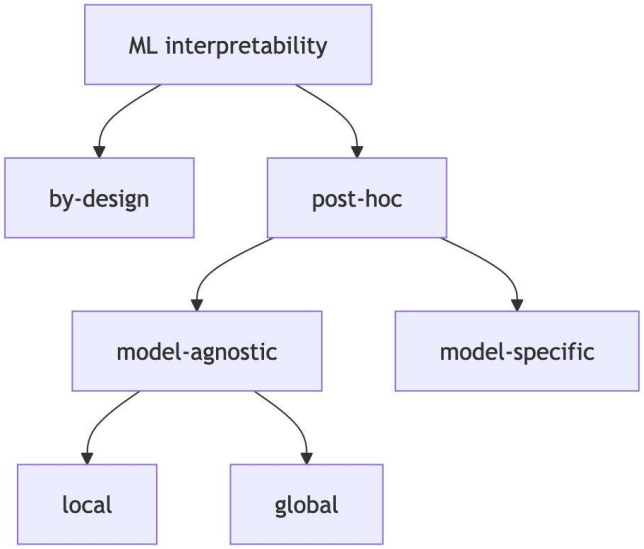

Figure 2.2 – Dimensions of Explainability in Neural Networks

2.1.3. Methods for Explainability in Neural Networks

Explainability methods for neural networks can be broadly categorized based on how they reveal model behavior, ranging from identifying important input features to generating human-readable rules or representative examples (Schwalbe & Finzel, 2024). Each method offers a different perspective on understanding complex model predictions, balancing depth of insight with practical usability.

One of the most common families of techniques is feature attribution methods, which provide a breakdown of how much each input feature contributed to a model's decision. These methods assign importance scores to features, helping users identify key factors in predictions, detect potential biases, and debug model behavior (Bach et al., 2015).

Moving beyond direct attribution, counterfactual and contrastive explanations explore how small changes to the input could alter the model's prediction. Instead of highlighting feature importance, these methods answer "what if?" questions, offering actionable insights into how different inputs could have led to a different outcome (Wachter et al., 2017).

Surrogate models offer another interpretability strategy by approximating the behavior of a complex neural network with a simpler, inherently interpretable model. By fitting an interpretable model to the black-box model's outputs, surrogate techniques enable users to

understand global or local decision patterns without altering the original architecture (Ribeiro et al., 2016).

Rule-based explanations extract explicit, human-readable rules from model behavior, describing decision boundaries through logical if-then statements. This structured format makes it easier to audit models, evaluate fairness, and ensure compliance, particularly in highly regulated industries (Lakkaraju et al., 2016).

Complementing these approaches, prototype and example-based explanations illustrate model reasoning through real-world representative cases rather than feature attributions. These methods explain decisions by linking predictions to similar known examples, supporting intuitive understanding (Kim et al., 2016).

Concept-based explainability shifts the focus from individual features to human-understandable concepts. By grouping related features into higher-level representations, these methods align model reasoning with domain knowledge, making explanations more meaningful (Koh et al., 2020).

In some cases, interpretability is built directly into the model design through architecturally-designed explainability. Models like self-explaining neural networks and neural additive models incorporate transparency into their structure, eliminating the need for separate post-hoc interpretation (Alvarez-Melis & Jaakkola, 2018).

Other methods prioritize visual clarity. Visualization-based explainability techniques, such as saliency maps and feature interaction plots, provide graphical insights by highlighting influential input regions or feature interactions. These visual tools are particularly useful for complex deep learning models where direct feature importance scores might be less intuitive (Simonyan et al., 2013).

A more causally focused perspective is provided by causal inference-based explainability, which seeks to identify cause-and-effect relationships rather than mere correlations. These methods aim to determine which factors directly impact an outcome, offering deeper insights into model behavior and actionable interventions (Pearl, 2000).

Finally, attention-based explainability methods leverage the attention mechanisms within models to highlight which parts of the input were most influential for a given prediction. Originally developed for natural language processing and computer vision, attention-based methods have also proven effective for explaining predictions in tabular data (Bahdanau et al., 2014).

Together, these diverse techniques form a rich toolkit for interpreting neural network predictions. Each method brings unique strengths depending on the specific context and operational requirements, highlighting the importance of carefully selecting appropriate explainability strategies for real-world applications.

2.2. STATE OF THE ART FOR EXPLAINABLE NEURAL NETWORKS IN TABULAR DATA

Building on the foundations of neural network explainability, this section reviews the state-of-the-art methods specifically developed for tabular data. The methods reviewed here are divided into two broad categories: intrinsic approaches, where models are designed to be interpretable by design, and post-hoc techniques, which aim to explain the behavior of already trained models.

2.2.1. Intrisic Explainability Methods

The growing complexity of machine learning models, especially deep neural networks, has posed significant challenges when it comes to understanding their decisions. In high-stakes domains like healthcare, finance, and telecommunications, it's essential to not only trust the accuracy of these models but also to comprehend the reasoning behind their predictions. Over the years, a range of intrinsically interpretable models and methods have emerged to address this challenge, each bringing new innovations to make models more interpretable.

The journey began with Self-Explaining Neural Networks (SENN), proposed by Alvarez-Melis and Jaakkola in 2018. In an era when most interpretability approaches were post-hoc, relying on tools like SHAP or LIME to explain model behavior after training, SENN took a bold step forward. By incorporating interpretability directly into the model's architecture, SENN aimed to make explanations inherent to the learning process. The authors introduced three key principles known as explicitness, faithfulness, and stability. These principles ensure that the model's decisions are understandable, consistent, and aligned with its true behavior. This development marked an important milestone by demonstrating that deep learning models can be both highly accurate and inherently interpretable.

In 2021, TabNet by Arik and Pfister further pushed the boundaries by combining the power of deep learning with interpretability. The key innovation was the use of sequential attention, which allowed the model to focus on the most relevant features for each instance. This feature selection was not performed after training but was an integral part of the model's architecture. As a result, TabNet not only performed well on tabular data but also provided clear and interpretable explanations for its decisions. Its success marked an important milestone by demonstrating that models could be both high-performing and transparent without needing to sacrifice one for the other.

That same year, Neural Additive Models (NAMs), proposed by Agarwal et al., offered a new approach by combining the best aspects of deep learning and Generalized Additive Models (GAMs). While deep learning models are often praised for their predictive power, they are typically criticized for being black-box models. NAMs addressed this by ensuring that each feature's contribution to the model's decision was independent and additive, allowing for clear, visual explanations of how each feature influenced the outcome. By introducing a novel

activation function (ExU units), NAMs could also model sharp, non-smooth feature effects, improving their flexibility while maintaining full interpretability. This was important because it bridged the gap between the power of deep learning and the transparency of simpler, more interpretable models like GAMs.

Building on the idea of GAMs, GAMI-Net by Yang et al. (2021) took the interpretability of neural networks even further by introducing structured pairwise interactions between features. While models like NAMs were additive, GAMI-Net allowed for meaningful interactions between features, making the model's predictions even more accurate while still ensuring interpretability. It was a crucial development for handling more complex data, where interactions between features are often key to making accurate predictions. GAMI-Net's focus on sparsity, heredity, and marginal clarity also helped maintain a clear structure in the learned relationships, which was essential for both model performance and interpretability.

As the complexity of models grew, the need for scalable and interpretable solutions became even more pressing. This need was addressed in 2022 with the introduction of Neural Basis Models (NBM) by Radenovic et al. NBM tackled one of the major limitations of previous models: scalability. While models like NAMs and Explainable Boosting Machines (EBMs) worked well on smaller datasets, they struggled with high-dimensional data. NBM reduced computational complexity by learning a small set of shared basis functions across features, which could be used to explain each feature's contribution to the model's prediction. This made NBM not only scalable to larger datasets but also interpretable, making it an important step forward for large-scale machine learning.

The same year, NODE-GAM by Chang et al. introduced a new approach by combining the benefits of GAMs with deep learning's scalability. NODE-GAM, based on the Neural Oblivious Decision Trees (NODE) architecture, enforced feature-wise additivity while leveraging the advantages of deep learning, such as differentiability and efficient optimization. This allowed NODE-GAM to handle large datasets without sacrificing transparency. Its ability to specialize each decision tree to a single feature, through temperature annealing, ensured that each feature's contribution remained interpretable, which was essential for applications where understanding the decision-making process is critical.

In 2023, LocalGLMnet by Richman and Wüthrich offered a significant advancement by integrating the interpretability of generalized linear models (GLMs) with the power of deep learning. The key innovation was the use of regression attentions, where the network learned instance-specific regression coefficients. This meant that, unlike traditional deep learning models, the relationships between features remained interpretable, and feature interactions could be explicitly analyzed. LocalGLMnet's ability to provide transparent regression analyses without relying on post-hoc explanations made it particularly valuable for domains requiring a clear understanding of how decisions were made.

That same year, Tabular Concept Bottleneck Models (TabCBM) by Espinosa Zarlenga et al. introduced a concept-based approach to explainability. While concept-based explanations had been explored for images and other structured data, TabCBM brought this idea to tabular data. The model learned a set of high-level concepts, which were defined by subsets of correlated features, allowing the model's predictions to be interpreted in a human-understandable way. This was a significant step forward, as it showed that even in highly structured data, meaningful concepts could be discovered that would enhance model transparency.

Another important method introduced in 2023 was SHAPNN by Cheng et al. This method integrated Shapley value-based regularization directly into the training of neural networks, providing real-time feature importance estimation. Unlike traditional post-hoc methods like SHAP, SHAPNN ensures that feature selection and evaluation are part of the model's learning process from the very beginning. This integration of Shapley values into training made SHAPNN a powerful tool for improving both the performance and transparency of deep neural networks, especially in dynamic environments where data distributions change over time.

LEURN (Learning Explainable Univariate Rules with Neural Networks), introduced by Aytekin in 2023, focused on the importance of simplicity in explainability. LEURN proposed a model that enforced an additive structure, with each rule contributing linearly to the decision-making process. Unlike traditional deep learning models that often rely on complex feature interactions, LEURN provided a simple, interpretable decision process by learning univariate decision rules. This simplicity made LEURN a valuable tool for applications requiring exact and interpretable rules.

In 2024, InterpreTabNet, an extension of TabNet, improved upon its predecessor by addressing the issue of overlapping attention masks. By regularizing the attention masks to promote sparsity, InterpreTabNet not only improved the interpretability of the model but also introduced a post-hoc natural language explanation step using GPT-4. This allowed for more intuitive and human-readable explanations, making the decision-making process even more transparent.

SEE-Net (Synced Explanation-Enhanced Neural Network), also proposed in 2024 by Seo and Li, took a unique approach by integrating a guiding deep neural network (DNN) with a shallow interpretable network. This hybrid architecture maintained high predictive performance while ensuring transparency. SEE-Net's use of co-supervision allowed the DNN to guide the training of an interpretable model, facilitating a trade-off between accuracy and interpretability. This was a key advancement, especially in scenarios where both performance and interpretability were required.

Also in 2024, Interpretable Mesomorphic Networks (IMN), proposed by Kadra et al., introduced deep hypernetworks that generated instance-specific linear models. This allowed the model to be both deep and locally linear, ensuring that predictions remained transparent

and faithful to the learned decision boundary. IMN's ability to provide transparent decision functions in the original feature space made it a powerful tool for applications where understanding individual decisions was crucial.

At the same time, IGNNet (Interpretable Graph Neural Network for Tabular Data), proposed by Alkhatib et al., introduced a graph-based approach to tabular data. By representing features as nodes in a graph, IGNNet was able to provide more transparent predictions, ensuring that each feature's contribution could be explicitly traced. This method not only performed well on tabular data but also provided true Shapley value-based explanations, a key advancement for making graph neural networks more interpretable.

GP-NAM (Gaussian Process Neural Additive Models), introduced by Zhang et al. in 2024, further enhanced NAMs by incorporating Gaussian processes to replace neural shape functions. This innovation allowed GP-NAM to reduce the number of parameters required, improving both efficiency and interpretability. It also introduced a more stable and smooth feature effect visualization, making it an efficient and interpretable alternative to standard NAM-based architectures.

Finally, GGNAM (Generalized Groves of Neural Additive Models) by Chen and Ye in 2024 addressed the limitations of NAMs in capturing feature interactions by introducing a framework that categorizes features into linear, nonlinear, and interacted components. This separation of features allowed GGNAM to offer greater transparency while maintaining high predictive accuracy, making it particularly useful for applications where interpretability is critical.

2.2.2. Post-hoc Explainability Methods

While intrinsically interpretable models offer transparency from the outset, the increasing complexity of deep learning models has also led to the development of post-hoc explainability methods. These techniques provide transparency and insights into model behavior after the model has been trained. Post-hoc methods aim to shed light on black-box models, allowing users to understand why a model made a particular prediction without altering the model's internal structure.

One of the earliest and most foundational methods in this area is LIME (Local Interpretable Model-Agnostic Explanations), proposed by Ribeiro et al. in 2016. LIME works by approximating black-box models with simpler, interpretable models locally around a specific prediction. This local fidelity ensures that the explanation matches the behavior of the model near the instance being explained. By generating perturbations of the input and observing the resulting model predictions, LIME fits a surrogate model, usually a sparse linear model or decision tree, to capture the model's behavior. This method not only helped establish a framework for interpreting complex models but also introduced the idea that model

explanations do not have to be global. They can be local, focusing on the instance at hand. LIME's ability to enhance user trust in predictions and improve decision-making was demonstrated across tasks such as text and image classification.

Building on the need for more reliable and consistent attributions, Lundberg and Lee introduced SHAP (SHapley Additive exPlanations) in 2017, offering a major leap in post-hoc interpretability. SHAP combines concepts from cooperative game theory to assign a value to each feature based on its contribution to the prediction. Unlike earlier methods, SHAP guarantees three key properties: local accuracy, missingness, and consistency. The method's Shapley values ensure that each feature is attributed fairly, reflecting its true contribution to the output. To make SHAP practical for large models, the authors also introduced Kernel SHAP, an approximation technique using weighted linear regression. SHAP's theoretical guarantees set it apart from methods like LIME, making it a preferred tool for more robust and interpretable machine learning.

Around the same time, another crucial method was developed: DeepLIFT (Deep Learning Important Features), introduced by Shrikumar et al. (2017). DeepLIFT's contribution lies in how it handles attribution in deep neural networks, particularly in networks that use non-linear activations like ReLU. Unlike gradient-based methods, which can suffer from vanishing gradients, DeepLIFT compares activations to a reference input, allowing for more stable feature attributions. This approach also avoids issues with gradients in saturated regions, making it more reliable for networks that use ReLU activations. Its ability to generate robust attributions without relying on gradients made DeepLIFT a powerful alternative to previous methods, particularly in deep learning contexts.

Further advancing the field, Integrated Gradients (IG), also introduced in 2017, takes the concepts of DeepLIFT a step further. The key contribution of IG, as proposed by Sundararajan et al., was the introduction of two axioms: Sensitivity and Implementation Invariance. These axioms ensured that attributions are consistent and meaningful, even across different model architectures. IG computes attributions by integrating gradients along a path from a baseline input to the actual input, providing a cumulative view of each feature's contribution. This method addressed several shortcomings of previous gradient-based approaches, especially in handling the complexities of deep neural networks.

In 2018, Ribeiro et al. continued their work on model-agnostic explanations with Anchors. This method differs from LIME by offering high-precision, if-then rule-based explanations that guarantee faithfulness to the original model. Anchors are designed to hold with high probability in the local region around the prediction, ensuring that users can trust the explanation. This high precision and clarity in rule-based explanations made Anchors particularly valuable in applications where users needed strong, actionable insights into model decisions. Compared to LIME, Anchors provided more stable and interpretable explanations, which was a crucial step toward building trust in complex machine learning systems.

Further enhancing the toolbox for model-agnostic explanations, Guidotti et al. (2018) introduced LORE (Local Rule-Based Explanations), a method that combines decision rules with counterfactuals. LORE generates synthetic neighborhoods around target instances using genetic algorithms, then applies decision trees to approximate the local decision boundary. The method provides compact, reliable decision rules that explain why a model made a specific prediction. Unlike LIME, which uses linear models, LORE captures nonlinear relationships and logical feature dependencies, making it a more powerful tool in certain contexts.

As the demand for more practical and actionable explanations grew, methods for counterfactual explanations began to gain traction. Mothilal et al. (2020) proposed DiCE (Diverse Counterfactual Explanations), a framework that generates multiple, diverse counterfactuals for a given prediction. Rather than providing just a single explanation, DiCE offers a set of plausible, actionable alternatives, giving users a broader range of insights into how to alter a model's outcome. The method's ability to optimize for diversity, while preserving the feasibility and proximity of the counterfactuals, made it an important development in making machine learning explanations more useful in real-world decision-making.

A few months later, Poyiadzi et al. (2020) introduced FACE (Feasible and Actionable Counterfactual Explanations), a method designed to ensure that counterfactuals are not just plausible but also realistic and implementable. FACE improves upon earlier counterfactual methods by ensuring that the suggested changes are actionable and achievable, particularly in domains where unrealistic suggestions could lead to unethical or impractical outcomes. By introducing high-density path-based counterfactual generation, FACE guarantees that the counterfactuals are situated in regions of the data distribution that make sense in the real world, which added an important layer of practicality to counterfactual explanations.

The latest frontier in surrogate-based explainability came with Engel et al.'s work on Neural Tangent Kernels (NTK) in 2024. NTK approximates the behavior of deep neural networks using kernel-based methods, providing high-fidelity local explanations without sacrificing model accuracy. The introduction of trace neural tangent kernels (trNTK) and their computational efficiency marked a significant step forward in creating scalable and interpretable surrogate models for deep learning. This method not only enabled faithful explanations but also allowed for a deeper understanding of how training data influenced the model's predictions.

Each of these methods represents a step forward in making machine learning models more interpretable. From generating local explanations with LIME and SHAP to the development of counterfactuals through DiCE and FACE, these techniques have provided valuable insights into how and why models make decisions.

2.3. SUMMARY AND RESEARCH GAPS

This literature review has explored the growing field of explainable neural networks, with a focus on applications to tabular data. It began by outlining the strengths of neural networks in modeling complex, high-dimensional relationships, while also acknowledging the persistent challenge of interpretability, particularly in high-stakes and regulated domains such as telecommunications. To address this, a broad range of explainability methods was examined, including both intrinsic architectures and post-hoc techniques such as feature attribution, counterfactual reasoning, surrogate modeling, and concept-based explanations. Table 2.1 provided a comparative overview of these methods, evaluating their scope, interpretability mechanisms, and ease of implementation.

Table 2.1 – Comparison of State of the Art Explainability Methods for Neural Networks

Method	**Category**	**Model Scope**	**Type of Explainability**	**Ease of Use**
TabNet	Intrisic	Specific	Architecturally-Designed	Easy
InterpreTabNet	Intrisic	Specific	Architecturally-Designed	Difficult
SEE-Net	Intrisic	Specific	Architecturally-Designed	Difficult
IMN	Intrisic	Specific	Architecturally-Designed	Moderate
LocalGLMnet	Intrisic	Specific	Architecturally-Designed	Moderate
SENN	Intrisic	Specific	Architecturally-Designed	Moderate
SHAPNN	Intrisic	Specific	Architecturally-Designed	Difficult
LEURN	Intrisic	Specific	Architecturally-Designed	Difficult
IGNNet	Intrisic	Specific	Architecturally-Designed	Moderate
NAMs	Intrisic	Specific	Architecturally-Designed	Moderate
GAMI-Net	Intrisic	Specific	Architecturally-Designed	Moderate
NBM	Intrisic	Specific	Architecturally-Designed	Moderate
NODE-GAM	Intrisic	Specific	Architecturally-Designed	Moderate
GP-NAM	Intrisic	Specific	Architecturally-Designed	Moderate
GGNAMs	Intrisic	Specific	Architecturally-Designed	Difficult

TabCBM	Intrisic	Specific	Architecturally-Designed	Moderate
SHAP	Post-hoc	Agnostic	Feature Attribution	Easy
LIME	Post-hoc	Agnostic	Feature Attribution, Surrogate Model	Easy
IG	Post-hoc	Specific	Feature Attribution	Easy
Anchors	Post-hoc	Agnostic	Rule-based, Surrogate model	Easy
DeepLIFT	Post-hoc	Specific	Feature Attribution	Easy
DiCE	Post-hoc	Agnostic	Counterfactual and Contrastive	Easy
FACE	Post-hoc	Agnostic	Counterfactual and Contrastive	Difficult
LORE	Post-hoc	Agnostic	Rule-based	Moderate
NTK	Post-hoc	Specific	Surrogate Model	Difficult

While explainable artificial intelligence (XAI) has gained significant attention, research on explainability in neural networks for tabular data remains underdeveloped. Most existing studies focus on deep learning models for image and text data, leaving a gap in understanding how explainability techniques affect performance in tabular data settings. In telecommunications, where predictive models are used to detect service issues, ensuring both high predictive performance and model interpretability is crucial. However, there is often a trade-off between explainability and predictive accuracy, particularly in deep learning models.

Several key research gaps emerge from the literature. One important gap concerns the trade-off between performance and interpretability in neural networks for tabular data. While simpler models, such as decision trees and linear models, are inherently interpretable, they often underperform compared to more complex neural network architectures. Conversely, neural networks tend to achieve superior predictive performance but at the cost of lower transparency. Despite its practical importance, there is limited research quantifying this trade-off in the context of tabular data, particularly for anomaly detection in telecommunications. This thesis aims to explore and systematically analyze this balance.

A second critical gap is the lack of practical evaluation of interpretability in real-world tabular settings. Although many explainability frameworks have focused on theoretical foundations or evaluations on standard benchmark datasets, few studies have assessed interpretability in

operational, domain-specific contexts. In telecommunications, where model transparency is vital for troubleshooting and proactive issue resolution, there remains a need to evaluate how interpretable a model truly is in practice, and whether achieving higher interpretability compromises predictive performance. To address this, this thesis will conduct an empirical comparison of explainable neural network methods, analyzing their trade-offs in terms of performance, fidelity, and practical usability in anomaly detection scenarios.

By addressing these gaps, this research aims to provide a structured analysis of the explainability-performance trade-off in neural networks for tabular data, with a specific focus on telecommunications anomaly detection and proactive issue resolution.

3. METHODOLOGY

This study adopts the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology as the foundation for structuring the research. CRISP-DM provides a systematic and iterative framework, supporting a logical progression from business understanding to model evaluation. Given the objective of developing an explainable neural network model for the proactive detection of technical issues in telecommunications, this methodology offers a particularly suitable guide for the research process.

Initially developed by Chapman et al. (2000), CRISP-DM is a widely recognized industry standard for machine learning and data mining projects. It consists of six phases. The first phase, Business Understanding, focuses on defining the problem, establishing objectives, and identifying constraints. The second phase, Data Understanding, involves collecting and exploring the dataset to assess its structure and quality. In the third phase, Data Preparation, the data is preprocessed, cleaned, and transformed to create a suitable format for modeling. The fourth phase, Modeling, involves selecting and applying appropriate machine learning techniques. The Evaluation phase follows, assessing whether the model meets the defined objectives. Finally, the Deployment phase focuses on implementing the model into real-world systems. A visual representation of the CRISP-DM process is provided in Figure 3.1.

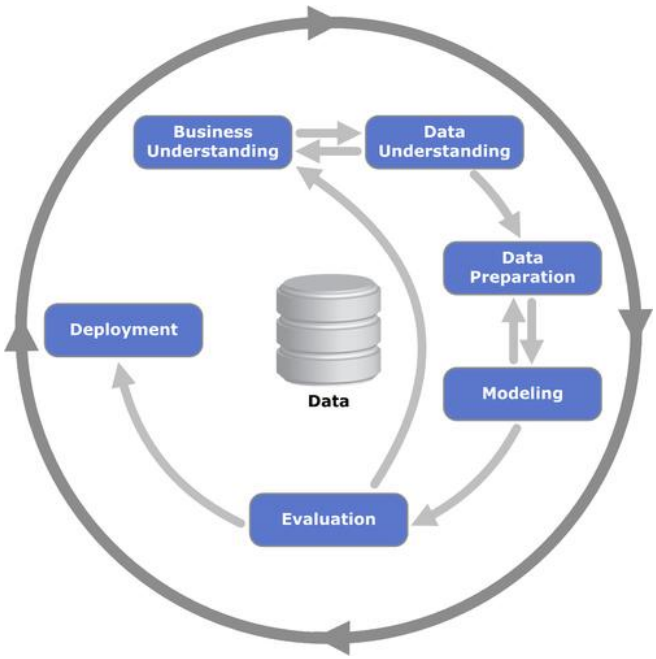

Figure 3.1 – CRISP-DM Process Overview

CRISP-DM is ideal for this study as it emphasizes understanding the problem before model development. Given the goal of proactively detecting technical issues, aligning the model with

telecommunications operations is crucial. The Business Understanding phase ensures clear objectives, while Data Understanding and Preparation enable efficient handling of large-scale, high-dimensional data. The iterative nature of CRISP-DM allows for continuous refinement, balancing predictive performance with explainability. As the focus is on model development and interpretability, the Deployment phase will not be addressed in this study.

3.1. BUSINESS UNDERSTANDING

In the telecommunications industry, customer satisfaction plays a crucial role in retention and profitability. Given the intense competition, even minor service disruptions can lead customers to switch providers, resulting in churn. Since acquiring new customers is more expensive than retaining existing ones, reducing churn is a key priority.

Traditional customer support methods are reactive, addressing issues only after customers report them. However, many customers silently experience technical problems without contacting support, leading to silent churn. This type of churn is particularly problematic because it often goes undetected until the customer leaves, causing lost revenue and diminished trust.

For NOS, transitioning from a reactive to a proactive issue resolution approach is critical. This study explores how proactive issue detection, using explainable neural networks, can help identify hidden technical problems before they escalate. By detecting issues early, NOS can reduce churn, improve customer satisfaction, and enhance service quality. The goal is to empower NOS with the ability to act preventively, addressing problems before they impact the customer experience.

3.2. DATA UNDERSTANDING

The dataset used in this study was provided by NOS, a leading telecommunications provider in Portugal. It consists of historical network telemetry and customer interaction data associated with customers and their equipment, collected during the months of April and May 2024. The raw data comprises approximately 78 million records and 447 features, covering a broad range of technical measurements, service events, and behavioral indicators.

However, not all records and features were suitable for training a supervised model. To build a clean and representative training set, both a conservative labeling framework and a careful feature selection process were applied. These preprocessing steps are explained in the following sections.

3.2.1. Label

The goal of this research is to proactively detect when a customer is experiencing a technical issue, specifically, when they are not only affected by degraded service but are also aware of the problem and dissatisfied with it. Since direct measurement of customer satisfaction is not scalable, this study uses Ordem de Trabalho (OTs), which is a record in the customer service system indicating a scheduled intervention or service issue, as a proxy for customer dissatisfaction. A customer-initiated OT is considered a strong indicator that the customer perceived a service issue and took action to resolve it.

The original labeling method, illustrated in Figure 3.2, was based on whether a customer had an OT scheduled within the next seven days from a given observation point. If an OT was scheduled within this window, the label was set to 1 (positive); if not, it was set to 0 (negative). Although straightforward, this approach introduced several weaknesses. First, the seven-day window was arbitrary and not necessarily aligned with real customer perception of technical issues. Second, the method did not account for potential data delays, which could lead to leakage or incorrect labeling. Finally, the negative labels were often noisy, as they could include ambiguous or unrelated interactions that did not reflect the true absence of technical issues.

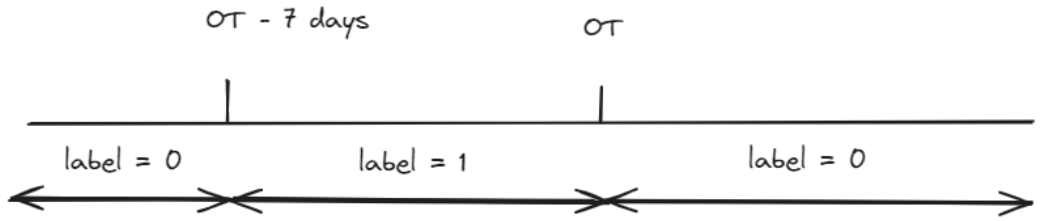

Figure 3.2 – Initial Labeling Strategy

To improve label quality, a more conservative, rule-based strategy was implemented. In this revised framework, a positive label (Label = 1) is assigned two days after a customer-initiated OT, accounting for potential data delays and ensuring that the label reflects the state of the network at the time the customer was likely experiencing the issue. A negative label (Label = 0) is assigned two days after a non-technical interaction, but only if there are no OTs or PTs (technical complaints) recorded within a ±17-day window around the interaction. This restriction ensures a high level of confidence that the customer was not experiencing technical issues. In all other cases, including ambiguous interactions, overlapping technical events, or periods with no interaction at all, the label is set to NULL. These records are excluded from model training to maintain label integrity.

Additional refinements were applied to the labeling logic. OTs triggered by internal systems are not considered positive labels, although they do create buffer zones to prevent

contamination. Certain interaction types, such as outbound calls or product changes, are excluded from negative labeling due to their lack of informative value regarding technical dissatisfaction. Furthermore, buffer zones of either ±17 or ±34 days are enforced to avoid overlapping events that could compromise the labeling process.

This revised approach prioritizes label quality over quantity, ensuring that the model learns only from high-confidence examples. As a result, the final labeled dataset includes approximately 220,000 records with either a positive or negative label. The complete labeling logic is summarized in Figure 3.3.

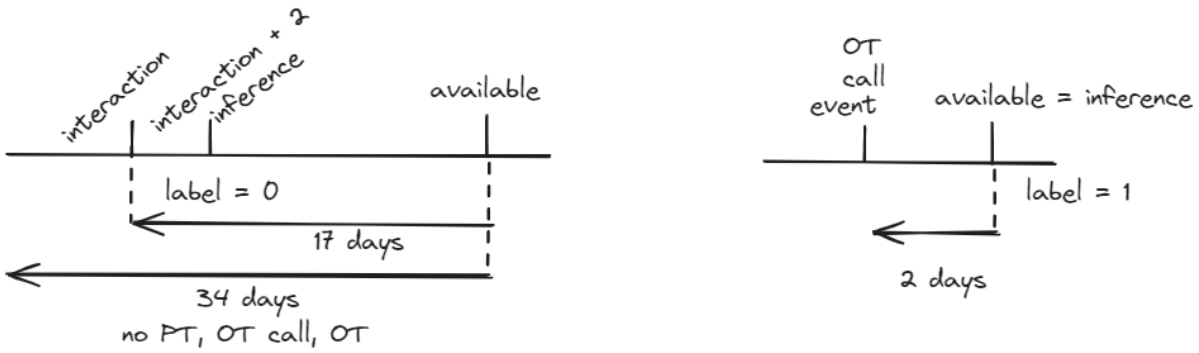

Figure 3.3 – Final Labeling Strategy

3.2.2. Feature Selection and Composition

Although the original dataset contained 447 features, only 248 were retained for modeling. This reduction was guided by a combination of factors, including domain expertise, business relevance, data quality, sparsity, and interpretability. Features that were administrative, redundant, weakly populated, or difficult to interpret were excluded. Additionally, one feature was excluded due to its potential to introduce data leakage, as it contained information that would not be available at prediction time.

Automated feature selection was also tested using feature importance scores from tree-based models such as Random Forest. However, this approach led to a degradation in model performance, and was therefore not adopted in the final feature selection process.

Each row in the final dataset represents a snapshot of customer-device behavior, anchored to a specific reference day. The feature set is composed primarily of numerical variables derived from network telemetry, capturing aspects such as device performance, signal quality, and network stability. Many of these features follow a structured naming convention, where the same metric is calculated over different time windows (e.g., 3, 7, 14, or 30 days), allowing the model to learn both short-term fluctuations and long-term degradation patterns. These include indicators of signal strength, bit error rates, device uptime, Wi-Fi stability, memory and CPU usage, and various anomaly rates. The majority of features were automatically

generated from internal monitoring systems and selected for their potential to act as early warning signals of customer-perceived technical issues. A complete list of the final features and their descriptions is provided in Appendix A.

3.3. DATA PREPARATION

The Data Preparation phase focused on transforming the raw labeled dataset into a clean and structured format suitable for supervised machine learning. Given the high dimensionality and technical nature of the data, this step included handling missing values and encoding categorical variables to ensure the final dataset was fully numerical and ready for modeling.

3.3.1. Handling Missing Values

The first step involved analyzing the presence and distribution of missing values. Null counts and percentages were calculated for all features, revealing that the vast majority of variables contained more than 30% missing values, with some exceeding 90%. After consulting with domain experts at NOS, it was determined that for numerical variables, missing values represented the absence of signal or event data and should therefore be interpreted as zeros. Based on this insight, all missing values in numerical features were imputed with 0.

The dataset also included two categorical variables: model_segment_id_tv and model_segment_id_net. In these cases, missing values indicated that the customer did not possess the corresponding equipment. For instance, a missing value in model_segment_id_tv meant the customer did not have a television box. These nulls were replaced with the string "Unknown" to preserve this semantic meaning.

No features were removed due to missingness, as the applied imputation strategy allowed retention of the entire feature set without introducing significant noise.

3.3.2. Categorical Encoding

After imputation, the two categorical features were one-hot encoded to ensure compatibility with machine learning models. This transformation expanded each feature into a set of binary columns, with clear prefix naming to maintain interpretability. As a result, all features in the dataset were converted to a fully numerical format.

3.3.3. Outlier Handling

No outlier removal was performed, as the features represent aggregated telemetry data and are expected to naturally exhibit a wide range of variation.

3.3.4. Feature Scaling

Feature scaling was applied during data preparation to ensure that all numerical features were on a comparable scale. This step is particularly important for models that are sensitive to feature magnitudes, such as the neural networks used in this study. The StandardScaler from the scikit-learn library was used, which transforms each feature by subtracting the mean and dividing by the standard deviation. As a result, each scaled feature has a mean of zero and a standard deviation of one. This normalization process improves numerical stability during training and ensures that all features contribute proportionally to the learning process.

3.3.5. Final Dataset State

Following these preprocessing steps, the dataset consisted of approximately 220,000 labeled records and 258 fully numerical features. The dataset was now clean, structured, and fully prepared for modeling.

3.4. MODELING

This section outlines the modeling phase of the study, including dataset splitting, benchmark model training, model selection, and neural network implementation. The objective is to develop and evaluate models capable of predicting whether a customer is experiencing and perceiving a technical issue, based on tabular network telemetry and interaction data. Both intrinsically interpretable neural networks and black-box neural networks with post-hoc explainability are considered, enabling a comprehensive analysis of the trade-off between predictive performance and interpretability.

3.4.1. Dataset Splitting

To simulate real-world deployment and prevent data leakage, the dataset was split using a temporal approach, where training, validation, and test sets were drawn from chronologically ordered customer-device snapshots. This strategy ensures that the model is trained on earlier data and evaluated on later data, accurately reflecting its ability to generalize to future, unseen cases.

The splitting procedure began by converting the available_day_id column, which represented the reference date for each observation, from its original integer format (e.g., 20240415) into a proper datetime format. The dataset was then sorted in ascending order by date to maintain temporal consistency. Once ordered, the first 75% of the records were allocated for training, the next 12.5% were used for validation, and the final 12.5% were reserved for testing. After the split, the date column was dropped from each subset, as it was not used as an input feature during the modeling phase.

These steps were applied to the full labeled dataset described in the previous section, which captures customer-device behavior across April and May 2024. The resulting split sizes were approximately 160,902 records for the training set, 26,817 records for the validation set, and 26,817 records for the test set. The label distribution across the splits remained moderately imbalanced, with approximately 23% to 25% of the records labeled as positive and 75% to 77% labeled as negative across all subsets. A visual summary of the temporal split is shown in Figure 3.3.

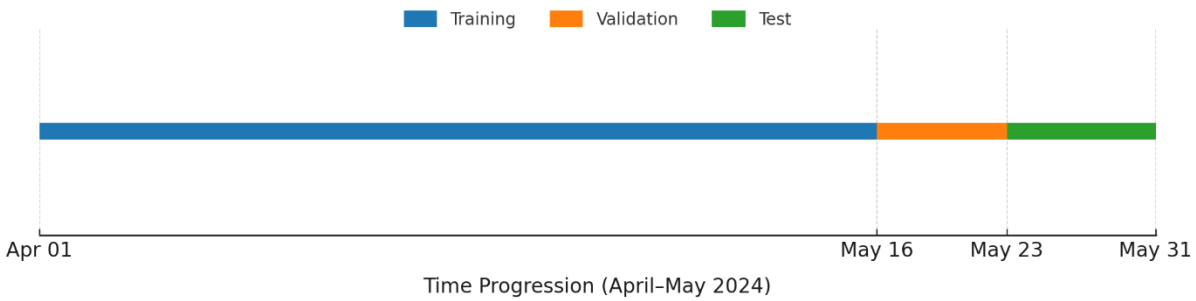

Figure 3.4 – Temporal Dataset Splitting Strategy

3.4.2. Benchmark Models

To establish a strong and interpretable benchmark, two state-of-the-art models were trained using the same training, validation, and test splits applied throughout the study: XGBoost and the Explainable Boosting Machine (EBM).

XGBoost is a high-performance, tree-based ensemble method widely recognized for its effectiveness on structured tabular data (Chen & Guestrin, 2016). It handles missing values natively, captures nonlinear relationships, and offers efficient training with built-in support for early stopping. As a leading model for tabular data, XGBoost provides a reliable baseline for comparison against more complex architectures. In this study, the XGBoost model was trained using a binary logistic objective, with early stopping based on validation set performance to prevent overfitting. Although it is not the primary model of interest, XGBoost serves as a robust reference point for evaluating the performance of the final neural network.

To complement the XGBoost benchmark, the Explainable Boosting Machine (EBM) was also employed as a second benchmark model. EBM is recognized for combining state-of-the-art predictive performance with high interpretability, particularly when applied to structured tabular datasets (Nori et al., 2019). As an inherently interpretable model from the interpret.glassbox library, EBM provides transparent insights into the relationships between input features and predictions, making it particularly useful for validating the results obtained from the neural network model. The EBM model was trained using the same data splits as XGBoost to ensure consistency across evaluations. By offering strong predictive capabilities alongside interpretability, EBM serves as a valuable benchmark for assessing both the performance and the explainability of the final neural network developed in this research.

3.4.3. Model Selection

The model selection process was based on both the literature review and practical considerations, particularly regarding ease of use, which refers to the availability of implementations and the quality of library support. For intrinsic explainability, the models chosen were TabNet, NAM, IMN, LocalGLMNet, SENN, IGGNet, GAMI-Net, NBM, NODE-GAM, GP-NAM, and TabCBM. While the implementation was not straightforward, the availability of pre-existing code made the models feasible to use.

For post-hoc explainability, a neural network was implemented using TensorFlow and Keras. The architecture included fully connected layers with ReLU activations, BatchNormalization, and Dropout to ensure robustness and prevent overfitting. Early stopping was used during training to optimize model performance. After training, several post-hoc techniques were applied, including SHAP, LIME, IG, Anchors, DeepLift, and DiCE. These methods were also chosen based on the literature review and the strong support available in libraries.

3.5. EVALUATION

The evaluation of the models in this study will focus on two main aspects: performance evaluation and interpretability evaluation. These two types of evaluations are essential for providing a comprehensive understanding of the model's effectiveness in identifying technical issues and ensuring that the model's decisions are transparent and understandable. Performance evaluation will assess how accurately the models predict customer technical issues, while interpretability evaluation will examine how well the models can explain their decision-making process. The balance between these two aspects is crucial for ensuring both the reliability and trustworthiness of the models in real-world applications.

3.5.1. Performance Evaluation

In this section, the focus is on evaluating the models' performance in predicting customer technical issues. Performance evaluation is essential to ensure that the models not only achieve strong predictive accuracy but also effectively identify customers who are likely experiencing technical problems, particularly given the class imbalance present in the dataset.

Several metrics are used to assess model performance. Accuracy provides a general measure of correct predictions across all classes and is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP denotes true positives, TN true negatives, FP false positives, and FN false negatives. While accuracy is widely used, it may not fully capture model effectiveness in imbalanced settings, and thus will be interpreted with caution. Precision measures the proportion of predicted positive cases that are actual positives and is given by:

$$Precision = \frac{TP}{TP + FP}$$

This metric is particularly important for minimizing false positives, ensuring that customers flagged as experiencing technical issues are indeed affected. Recall assesses the proportion of actual positive cases that the model correctly identifies and is computed as:

$$Recall = \frac{TP}{TP + FN}$$

High recall is critical to ensure that as many truly affected customers as possible are detected by the model. The F1-score, which is the harmonic mean of precision and recall, provides a balanced measure of performance and is calculated as:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

It offers a comprehensive view of the model's ability to balance precision and recall. The primary emphasis in this evaluation will be on precision for the positive class, as minimizing false positives is critical when identifying customers likely to experience technical issues. Recall and F1-score for the positive class will also be considered, but they will be treated as secondary metrics in the final assessment.

3.5.2. Interpretability Evaluation

Interpretability is a key factor in building trust in machine learning models, particularly in domains like telecommunications, where decisions can have significant consequences for

customers. In this section, the models' interpretability is evaluated by examining several important aspects related to how well their decisions can be understood and trusted.

The first criterion is consistency. Consistency refers to the stability of a model's interpretability outputs across similar inputs. In this study, consistency was evaluated using a k-nearest neighbors (k-NN) approach. For each of 50 randomly selected test instances, the k nearest neighbors (where k=3,5,10) were identified in the input space using cosine similarity. The interpretability outputs, such as feature importance scores or relevance weights, were then compared between each target instance and its neighbors by computing the cosine similarity of their explanation vectors. A high similarity score indicates that the model assigns similar importance patterns to similar inputs, thereby increasing trust in the model's internal reasoning (Alvarez-Melis & Jaakkola, 2018). A global consistency estimate for each model was obtained by averaging the similarity scores across the 50 sampled instances and reporting both the mean and standard deviation.

The second criterion is sparsity. Sparsity measures how concise an explanation is, specifically focusing on how many input features are needed to account for the majority of the model's explanation (Nauta et al., 2023). Two complementary sparsity metrics were used to assess this property. First, for each instance, the minimum number of features required to explain 90% of the total importance, based on feature attributions or relevance scores, was calculated. Lower values in this metric indicate more focused and concise explanations. Second, the coverage of the top 10 most important features was measured to determine how much of the total explanation is captured by a small subset of features. This helps evaluate whether the model's reasoning can be summarized by a compact and interpretable group of inputs. Both sparsity metrics were computed for 50 randomly selected test instances per model, and the mean and standard deviation were reported across the samples. High sparsity, reflected by fewer required features or higher top-k coverage, is desirable as it leads to more human-readable explanations and facilitates expert interpretation in operational settings.

The third and final criterion is computational cost. While achieving high performance and interpretability is important, it is also necessary to ensure that the model can be efficiently deployed without imposing excessive computational burdens, particularly in real-time or resource-constrained environments. In this study, computational cost was estimated by measuring the average time required to generate an explanation for a single instance and scaling it to 1,000 instances, which approximates the typical volume of customers processed during each proactive outreach cycle by NOS. This estimation provides a realistic sense of the operational burden associated with each explainability method. Models or methods with prohibitively high runtimes were considered unsuitable for large-scale deployment.

3.5.3. Human Evaluation of Interpretability

After filtering the top models based on performance and interpretability metrics, a human evaluation was conducted to assess the quality and practical value of the explanations generated by each method. This evaluation aimed to identify the most interpretable and trustworthy model from the perspective of domain experts, ensuring that the selected approach is not only technically sound but also usable in real-world telecommunications operations.

The evaluation involved eight domain experts from the Healing project, within which this research is embedded. The participants included professionals with expertise in network diagnostics, data analysis, and technical operations. Their practical knowledge of telecommunications challenges and customer support processes made them particularly well-suited to assess the quality of model explanations. Table 3.1 presents the distribution of the domain experts across the evaluation forms.

Table 3.1 – Overview of Domain Expert Roles by Evaluation Form

Form Number	Position
Form 1	Data Engineer
Form 2	Data Analyst
Form 3	Business Expert
Form 4	Data Engineer
Form 5	Product Owner
Form 6	Business Expert
Form 7	Product Owner
Form 8	Business Expert

Each participant was presented with model explanations structured in a clear and intuitive visual format. Each explanation was displayed as a horizontal bar chart, designed to highlight the features that most influenced the model's prediction. Every bar represented a feature, labeled with the feature name alongside its actual value in parentheses, and the length and color of the bar indicated the feature's contribution to the prediction. Color coding was employed to clarify the nature of the influence: red bars indicated features contributing toward predicting a technical issue, blue bars represented features contributing toward

predicting the absence of a technical issue, and green bars indicated the feature was considered important, but the direction of its influence was not determined, only its relevance. Examples of these explanation graphs are provided in Figures 3.3 and 3.4.

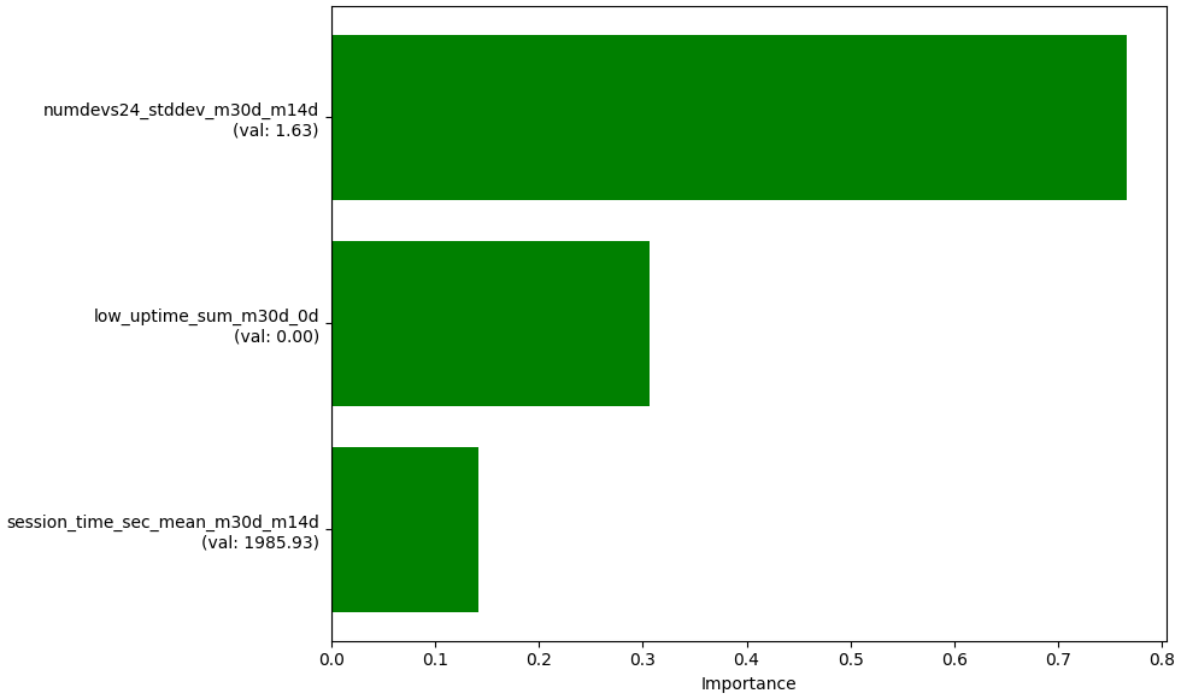

Figure 3.5 – TabNet Explanation Graph

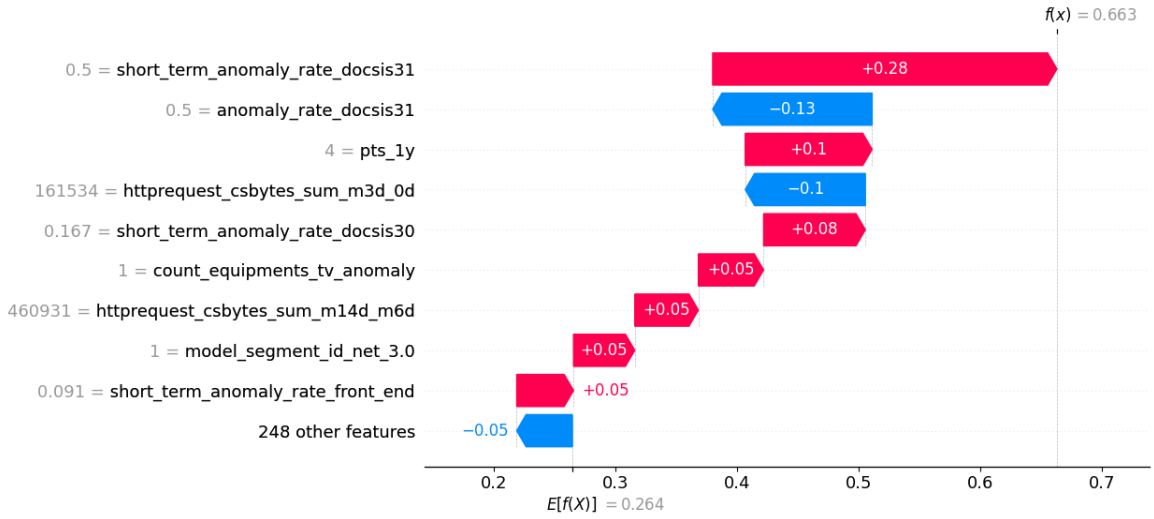

Figure 3.6 – SHAP Explanation Graph

The evaluation procedure was carefully structured to ensure fairness and consistency. Each participant received a customized evaluation form containing two explanations for each of the final selected methods. For every method, one explanation corresponded to a positive instance, meaning a customer experiencing a technical issue, and one to a negative instance, meaning a customer not experiencing any technical problems. The same two instances were used across all methods within each form, allowing for direct comparison. A total of eight different forms were generated, ensuring coverage of eight unique instance pairs, consisting of eight positive and eight negative examples from the test set. The explanation graphs were preprocessed and randomized using a script that ensured no two explanations from the same method appeared consecutively. This approach was designed to reduce potential evaluator bias and promote fairer comparisons between methods. Google Forms was used to deliver the evaluation, enabling remote access and structured response collection. All Google Form links used in the evaluation are provided in Appendix B.

The evaluation metrics were based on the human-grounded evaluation framework proposed by Doshi-Velez and Kim (2017) and further reviewed by Nauta et al. (2023). Participants were asked to assess each explanation using a 5-point Likert scale according to several criteria. The first criterion was understandability, measuring how easy it was to understand the explanation, rated from 1 (very difficult) to 5 (very easy). The second criterion was usefulness, evaluating how helpful the explanation was for understanding the model's behavior, also rated from 1 (not useful) to 5 (very useful). The third criterion was trustworthiness, assessing how much the participant trusted the explanation, ranging from 1 (do not trust) to 5 (completely trust).

Following these subjective assessments, participants completed a final task aimed at evaluating how effectively the explanation communicated the model's reasoning. They were asked to predict, based solely on the explanation, whether they believed the customer was experiencing a technical issue (yes or no). They also rated their confidence level in this prediction on a scale from 1 (not confident) to 5 (very confident). These combined metrics provided both subjective and objective insights into the effectiveness of each explanation, enabling a comprehensive comparison of interpretability across models.

The final model selection was based directly on the responses collected through the human evaluation forms. For each explanation method, the mean scores were computed across all participant responses for the subjective metrics: understandability, usefulness, trustworthiness, and confidence level. To assess the objective dimension of interpretability, participant predictions were compared both to the model's output and to the ground truth labels. Two key aspects were quantified. Faithfulness was measured as the percentage of participant predictions that matched the model's prediction, reflecting how accurately the explanation conveyed the model's reasoning. Correctness was measured as the percentage of participant predictions that matched the ground truth label, indicating whether the explanation supported accurate human decision-making.

The final decision on the most suitable model considered all of these metrics, both subjective and objective, alongside the business objectives of NOS. This ensured that the selected explanation method was not only interpretable and trustworthy from the user's perspective but also well-aligned with the operational needs of proactive customer support.

4. RESULTS AND DISCUSSION

This section presents the performance and interpretability evaluation of various models, focusing on their ability to detect customers experiencing technical issues in telecommunications. The models were assessed based on predictive performance, interpretability criteria, and feedback from domain experts. The following results provide insights into the strengths and trade-offs of each model in real-world applications.

4.1. PERFORMANCE RESULTS

Table 4.1 presents the predictive performance of benchmark models and state-of-the-art explainable neural networks, evaluated across multiple performance metrics. The main focus is on Label 1, which represents customers who are experiencing and perceiving technical issues, the primary target of this study.

Table 4.1 – Performance of Benchmark and Explainable Neural Network Models

Model	Precision (Label 1)	Recall (Label 1)	F1-Score (Label 1)	Precision (Label 0)	Recall (Label 0)	F1-Score (Label 0)	Accuracy
XGBoost	0,74	0,36	0,49	0,82	0,96	0,88	0,81
NBM	0,72	0,27	0,4	0,81	0,95	0,87	0,8
EBM	0,71	0,37	0,49	0,82	0,95	0,88	0,8
TabNet	0,71	0,31	0,43	0,8	0,96	0,87	0,79
Neural Network	0,71	0,3	0,42	0,8	0,96	0,87	0,79
LocalGLMNet	0,7	0,3	0,42	0,8	0,94	0,87	0,79
IMN	0,68	0,31	0,43	0,8	0,95	0,87	0,79
IGNNet	0,67	0,26	0,37	0,79	0,96	0,87	0,78
NAM	0,66	0,24	0,35	0,79	0,96	0,86	0,78

NODE-GAM	0,66	0,2	0,3	0,78	0,97	0,87	0,77
SENN	0,64	0,31	0,43	0,8	0,94	0,86	0,78
TabCBM	0,63	0,03	0,07	0,75	0,99	0,86	0,75
GAMI-Net	0,53	0,3	0,39	0,79	0,91	0,85	0,76
GP-GAM	0,27	0,03	0,06	0,75	0,97	0,84	0,73

To identify the most promising models for interpretability evaluation, candidates were filtered primarily based on precision for Label 1, which was the most important metric in this context. High precision is crucial to minimize false positives, ensuring that customers flagged as having technical issues are truly experiencing them.

Although recall and F1-score for Label 1 were also considered to capture broader model behavior, they served as secondary metrics in the selection process. Metrics related to Label 0 were not heavily weighted, as performance across models on the negative class was consistently strong. Similarly, overall accuracy was deemed satisfactory for all models under consideration and was not a decisive factor.

Based on this filtering process, five models were selected for interpretability evaluation. These models were the Neural Basis Model (NBM), TabNet, a Neural Network with post-hoc explainability techniques, LocalGLMNet, and Interpretable Mesomorphic Networks (IMN). Each of these models offered the best performance, demonstrating particularly strong precision for the positive class, and were therefore considered the most suitable candidates for deeper interpretability analysis.

The Neural Network (Post-hoc) model will be evaluated using a suite of widely adopted post-hoc explainability techniques, including SHAP, LIME, Integrated Gradients (IG), Anchors, DeepLIFT, and DiCE, as introduced in Section 3.4.3. These post-hoc methods, together with the selected intrinsically interpretable models NBM, TabNet, LocalGLMNet, and IMN will be analyzed in the following section to assess their interpretability characteristics, computational cost, and practical usability in real-world telecommunications scenarios.

4.2. INTERPRETABILITY RESULTS

As outlined in Section 3.5.2, interpretability was evaluated along three complementary dimensions: consistency, sparsity, and computational cost. These criteria were chosen to

reflect both theoretical and practical aspects of model transparency, with a particular focus on their relevance to real-world deployment in the telecommunications sector.

4.2.1. Computational Cost

Two post-hoc techniques, Anchors and DiCE, were excluded from further evaluation due to their high computational requirements. Both methods imposed significant runtime and memory overhead, making them impractical for large-scale application or real-time usage in operational environments. All remaining methods demonstrated acceptable computational performance and were therefore retained for detailed evaluation. Table 4.2 focuses on the consistency and sparsity results for all retained methods, reporting the mean and standard deviation for each quantitative metric.

Table 4.2 – Interpretability Evaluation of the Selected Explanation Methods

Method	**Explainability Type**	**Consistency (k=3)**	**Consistency (k=5)**	**Consistency (k=10)**	**Features 90% contribution (#)**	**Top 10 Coverage (%)**
SHAP	Post-hoc	0.85 ± 0.13	0.84 ± 0.12	0.82 ± 0.13	85 ± 11	41 ± 6
LIME	Post-hoc	0.84 ± 0.05	0.84 ± 0.06	0.84 ± 0.06	121 ± 4	26 ± 1
IG	Post-hoc	0.84 ± 0.13	0.83 ± 0.12	0.81 ± 0.13	96 ± 14	37 ± 7
DeepLIFT	Post-hoc	0.80 ± 0.17	0.80 ± 0.15	0.78 ± 0.16	94 ± 13	40 ± 10
TabNet	Intrisic	0.62 ± 0.27	0.61 ± 0.24	0.61 ± 0.21	4 ± 1	100
NBM	Intrisic	0.86 ± 0.1	0.83 ± 0.11	0.82 ± 0.12	81 ± 15	44 ± 6
LocalGLMNet	Intrisic	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	135 ± 10	50 ± 7
IMN	Intrisic	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	80 ± 11	43 ± 7

4.2.2. Post-hoc Methods

Among the retained post-hoc methods (SHAP, LIME, IG, and DeepLIFT), all exhibited satisfactory consistency across the evaluated test samples, with cosine similarity scores remaining relatively high for $k=\{3,5,10\}$ nearest neighbors. However, DeepLIFT consistently

yielded lower consistency scores compared to the other techniques, indicating less stable explanations across similar inputs.

In terms of sparsity, LIME required the highest number of features to explain 90% of the model's attribution and had the lowest top-10 feature contribution. This suggests that LIME produces more diffuse explanations, which may reduce interpretability in practical settings. SHAP and IG demonstrated a favorable balance between consistency and sparsity, offering both stable and relatively concise explanations. These two methods were therefore selected as the representative post-hoc techniques for further analysis.

4.2.3. Intrinsic Models

All intrinsically interpretable models, with the exception of TabNet, exhibited high consistency across similar input samples. TabNet's comparatively low consistency scores can be attributed to its instance-wise sparse feature selection mechanism, which leads to variable attention masks and, consequently, less stable attribution patterns across similar observations.

However, TabNet demonstrated exceptional sparsity performance, requiring, on average, four features to explain the full prediction and achieving 100% top-10 coverage. Given the importance of concise explanations for domain expert interpretation, this trade-off was deemed acceptable, and TabNet was retained for further evaluation.

Among the remaining intrinsic models, NBM was selected over IMN due to its slightly superior predictive performance, despite both models demonstrating similar interpretability characteristics. LocalGLMNet, while exhibiting excellent consistency and strong top-10 coverage, was excluded due to its relatively poor cumulative sparsity, requiring, on average, over 135 features to explain 90% of its predictions.

Taken together, the models and methods selected for the next stage of evaluation include NBM, TabNet, SHAP, and IG, as they represent the best trade-offs between interpretability quality, computational feasibility, and model performance.

4.3. HUMAN EVALUATION RESULTS

This section presents the results of the human evaluation conducted with domain experts. As described in Section 3.5.3, the evaluation was based on structured form responses and included both subjective assessments and objective prediction alignment, which together informed the final model selection. Table 4.3 summarizes the results across the four selected methods: IG, NBM, SHAP, and TabNet.

Table 4.3 – Human Evaluation of the Selected Explanation Methods

Method	Understandability	Usefulness	Trustworthiness	Confidence Level	Faithfulness (%)	Correctness (%)
IG	2,94	3,19	3,44	3,19	62	69
NBM	3,25	3,69	3,56	3,56	50	75
SHAP	2,81	3,12	3,25	3,31	56	69
TabNet	3,44	3,31	3,44	2,94	75	69

Among all methods, NBM achieved the highest ratings in usefulness, trustworthiness, and confidence level, with scores of 3.69, 3.56, and 3.56, respectively. This suggests that participants perceived its explanations as both reliable and actionable. TabNet ranked highest in understandability, scoring 3.44, which is consistent with its sparse and focused explanations. However, it scored the lowest in participant confidence, with a score of 2.94, indicating some hesitation when using its explanations to support decisions.

From an objective perspective, TabNet achieved the highest faithfulness at 75%, meaning participants' predictions most closely matched the model's outputs. In contrast, NBM achieved the highest correctness, also at 75%, indicating that its explanations were more effective in guiding participants to make accurate decisions when compared to the ground truth.

SHAP and IG performed slightly lower across both subjective and objective dimensions, although they still demonstrated balanced interpretability profiles.

These findings highlight meaningful trade-offs. TabNet excels in faithfully reflecting the model's logic, making it a strong choice for tasks where transparency into the model's reasoning is the top priority. NBM, on the other hand, supports more confident and accurate human decisions, making it particularly well-suited for operational use cases that prioritize usability and decision support.

Ultimately, both NBM and TabNet emerged as strong candidates, each excelling in different areas. The final model selection should therefore be guided by the intended goal, whether the focus is on faithfully conveying model logic or maximizing correct human judgment. Both methods offer valuable interpretability characteristics and can be applied based on the specific needs of proactive support at NOS.

5. CONCLUSIONS AND FUTURE WORKS

5.1. CONCLUSIONS

In this thesis, we investigated the potential of explainable neural networks for the proactive detection of customer-perceived technical issues in the telecommunications sector, using NOS as a case study. The primary objective was to bridge the gap between high predictive accuracy and model interpretability, two often competing goals, in order to improve customer satisfaction, reduce silent churn, and support operational decision-making.

The study began by contextualizing the importance of proactive issue resolution within the telecommunications landscape. Traditional reactive approaches were shown to be insufficient for addressing the needs of silent customers, who experience technical problems but do not report them. This insight formed the basis for the central research question: how can explainable neural network models be leveraged to proactively detect customer-perceived technical issues in telecommunications, and what is the trade-off between predictive accuracy and interpretability in these models when applied to tabular data?

To answer this question, the research followed the CRISP-DM framework, guiding the process through structured phases of business understanding, data preparation, modeling, and evaluation. A high-confidence labeled dataset was constructed from network telemetry and customer interaction data, applying conservative labeling rules designed to better reflect real-world customer perceptions. After extensive preprocessing, several neural network architectures, both intrinsically interpretable models and black-box models with post-hoc explainability, were developed and evaluated.

The results of the study provided several important insights. In terms of performance trade-offs, explainable neural networks such as NBM and TabNet delivered predictive performance comparable to benchmark models like XGBoost and EBM. This finding demonstrates that achieving explainability does not necessarily require sacrificing model effectiveness, and that interpretable models can be confidently considered for real-world deployment in proactive customer support settings.

Regarding intrinsic models, TabNet demonstrated the best sparsity, producing concise and focused explanations that participants found easy to understand. NBM, while not the most consistent overall, showed solid results across all interpretability metrics and offered the best trade-off between model performance and interpretability. Among post-hoc methods, SHAP and Integrated Gradients (IG) emerged as the most stable and informative techniques, offering reliable insights into model behavior.

The human evaluation, involving eight domain experts from NOS, provided additional insights into the interpretability of the selected models. TabNet explanations most faithfully reflected the model's predictions, while NBM explanations led to the highest number of correct human

predictions, indicating their value in guiding effective decision-making. Together, these results highlight the complementary strengths of both models and their relevance for different operational goals within proactive support at NOS.

Overall, the outcomes of this research validate the practical feasibility of deploying explainable neural networks for anomaly detection in telecommunications. More importantly, the study demonstrates that model interpretability does not have to come at the expense of predictive performance. Well-designed models can successfully balance transparency and operational utility, paving the way for more proactive and customer-centric technical support strategies.

5.2. FUTURE WORK

While this study makes important contributions to both research and practice, several avenues for future improvement and exploration have emerged.

One promising direction is the integration of natural language explanations using large language models (LLMs). While feature importance graphs provide structured insights into model reasoning, survey results indicated that these visual methods were not always intuitive or sufficient for all users. To address this, future work could leverage the existing Excel file containing the mapping between features and human-readable descriptions to generate textual summaries. By combining the output of feature importance methods with this predefined mapping, a language model such as OpenAI's GPT-4 API could produce concise, personalized explanations. For example, alongside the visual graph, the system might also generate a statement such as: "The customer's device has experienced a three-day drop in signal strength and increased error rates, likely leading to poor connectivity." This approach would improve accessibility for non-technical users and support more informed decision-making in customer-facing operations.

Another potential area for enhancement lies in expanding the group of human evaluators. In this study, the interpretability assessment involved eight domain experts, whose feedback provided valuable insights into the strengths and limitations of the models. However, a larger and more diverse pool of evaluators would allow for a more statistically robust evaluation. It would also offer a deeper understanding of how various profiles perceive and utilize model explanations in operational settings.

Finally, future research could explore the integration of real-time and continual learning capabilities. The models developed in this thesis were trained on historical data using a static labeling approach. A natural next step would be to enable the system to process live data streams and learn continuously from new information. This would allow the model to detect anomalies as they occur and adapt to changing network conditions or customer behavior over

time. Such improvements would better meet the operational needs of a live telecommunications environment, where timely detection and response are essential.

5.3. FINAL REMARKS

The findings of this thesis highlight the transformative potential of explainable neural networks in customer experience management for telecommunications. By proactively identifying and explaining technical issues before they result in churn, service providers like NOS can improve customer retention through faster and more transparent support. The balance between predictive performance and transparency is not only attainable, but also essential for the adoption of AI-driven systems in high-stakes domains.

The methodology, models, and evaluation frameworks presented in this work can serve as a blueprint for similar initiatives across the telecom industry and beyond, ultimately contributing to more transparent, customer-centric AI systems.

BIBLIOGRAPHICAL REFERENCES

Agarwal, R., Melnick, L., Frosst, N., Zhang, X., Lengerich, B., Caruana, R., & Hinton, G. E. (2021). Neural additive models: Interpretable machine learning with neural nets. Advances in neural information processing systems, 34, 4699-4711.

Al-Mashraie, M., Chung, S. H., & Jeon, H. W. (2020). Customer switching behavior analysis in the telecommunication industry via push-pull-mooring framework: A machine learning approach. Computers & Industrial Engineering, 144, 106476.

Al_Janabi, S., & Razaq, F. (2019). Intelligent big data analysis to design smart predictor for customer churn in telecommunication industry. In A. E. Hassanien, M. F. Tolba, A. T. Azar, & A. K. Darwish (Eds.), Big data and smart digital environment (pp. 246–272). Springer International Publishing.

Alkhatib, A., Ennadir, S., Boström, H., & Vazirgiannis, M. (2023). Interpretable graph neural networks for tabular data. arXiv preprint arXiv:2308.08945.

Alvarez Melis, D., & Jaakkola, T. (2018). Towards robust interpretability with self-explaining neural networks. Advances in neural information processing systems, 31.

Alvarez-Melis, D., & Jaakkola, T. S. (2018). On the robustness of interpretability methods. arXiv preprint arXiv:1806.08049.

Amin, A., Al-Obeidat, F., Shah, B., Adnan, A., Loo, J., & Anwar, S. (2019). Customer churn prediction in telecommunication industry using data certainty. *Journal of Business Research*, *94*, 290-301.

Angelov, P., & Soares, E. (2020). Towards explainable deep neural networks (xDNN). Neural Networks, 130, 185-194.

Arik, S. Ö., & Pfister, T. (2021, May). Tabnet: Attentive interpretable tabular learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 8, pp. 6679-6687).

Ascarza, E., Netzer, O., & Hardie, B. G. (2018). Some customers would rather leave without saying goodbye. Marketing Science, 37(1), 54–77.

Aytekin, C. (2023). LEURN: Learning Explainable Univariate Rules with Neural Networks. arXiv preprint arXiv:2303.14937.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one, 10(7), e0130140.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Barkai, O., & Harison, E. (2011). Preventive service management: Towards pro-active improvement of service quality. Review of Business Information Systems (RBIS), 15(4), 19-30.

Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. Electronics, 8(8), 832.

Chang, C. H., Caruana, R., & Goldenberg, A. (2021). Node-gam: Neural generalized additive model for interpretable deep learning. arXiv preprint arXiv:2106.01613.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide. SPSS Inc.

Chen, D., & Ye, W. (2024, October). Generalized Groves of Neural Additive Models: Pursuing Transparent Machine Learning Models in Finance. In 2024 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr) (pp. 1-8). IEEE.

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

Cheng, Q., Qu, S., & Lee, J. (2023). SHAPNN: Shapley Value Regularized Tabular Neural Network. arXiv preprint arXiv:2309.08799.

Cherukuri, H., Singh, S. P., & Vashishtha, S. (2020). Proactive issue resolution with advanced analytics in financial services. The International Journal of Engineering Research, 7(8), a1–a13.

Das, A., & Rad, P. (2020). Opportunities and challenges in explainable artificial intelligence (xai): A survey. arXiv preprint arXiv:2006.11371.

Dongare, A. D., Kharde, R. R., & Kachare, A. D. (2012). Introduction to artificial neural network. International Journal of Engineering and Innovative Technology (IJEIT), 2(1), 189-194.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608.

Engel, A., Wang, Z., Frank, N. S., Dumitriu, I., Choudhury, S., Sarwate, A., & Chiang, T. (2023). Faithful and efficient explanations for neural networks via neural tangent kernel surrogate models. arXiv preprint arXiv:2305.14585.

Espinosa Zarlenga, M., Shams, Z., Nelson, M. E., Kim, B., & Jamnik, M. (2024). Tabcbm: Concept-based interpretable neural networks for tabular data.

Fornell, C. (1992). A national customer satisfaction barometer: The Swedish experience. Journal of Marketing, 56(1), 6–21.

Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., & Giannotti, F. (2018). Local rule-based explanations of black box decision systems. arXiv preprint arXiv:1805.10820.

Haykin, S. (1994). Neural networks: a comprehensive foundation. Prentice Hall PTR.

Kadra, A., Pineda Arango, S., & Grabocka, J. (2025). Interpretable Mesomorphic Networks for Tabular Data. Advances in Neural Information Processing Systems, 37, 31759-31787.

Khatibi, A. A., Ismail, H., & Thyagarajan, V. (2002). What drives customer loyalty: An analysis from the telecommunications industry. Journal of Targeting, Measurement and Analysis for Marketing, 11, 34-44.

Kim, B., Khanna, R., & Koyejo, O. O. (2016). Examples are not enough, learn to criticize! criticism for interpretability. Advances in neural information processing systems, 29.

Kim, H. S., & Yoon, C. H. (2004). Determinants of subscriber churn and customer loyalty in the Korean mobile telephony market. Telecommunications policy, 28(9-10), 751-765.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., & Liang, P. (2020, November). Concept bottleneck models. In International conference on machine learning (pp. 5338-5348). PMLR.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016, August). Interpretable decision sets: A joint framework for description and prediction. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1675-1684).

Linardatos, P., Papastefanopoulos, V., & Kotsiantis, S. (2020). Explainable ai: A review of machine learning interpretability methods. Entropy, 23(1), 18.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in neural information processing systems, 30.

Molnar, C. (2020). Interpretable machine learning. Lulu. com.

Mothilal, R. K., Sharma, A., & Tan, C. (2020, January). Explaining machine learning classifiers through diverse counterfactual explanations. In Proceedings of the 2020 conference on fairness, accountability, and transparency (pp. 607-617).

Nauta, M., Trienes, J., Pathak, S., Nguyen, E., Peters, M., Schmitt, Y., ... & Seifert, C. (2023). From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. ACM Computing Surveys, 55(13s), 1-42.

Neuberg, L. G. (2003). Causality: models, reasoning, and inference, by judea pearl, cambridge university press, 2000. Econometric Theory, 19(4), 675-685.

Nori, H., Jenkins, S., Koch, P., & Caruana, R. (2019). Interpretml: A unified framework for machine learning interpretability. arXiv preprint arXiv:1909.09223.

Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., De Bie, T., & Flach, P. (2020, February). FACE: feasible and actionable counterfactual explanations. In Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (pp. 344-350).

Radenovic, F., Dubey, A., & Mahajan, D. (2022). Neural basis models for interpretability. Advances in Neural Information Processing Systems, 35, 8414-8426.

Rai, A. (2020). Explainable AI: From black box to glass box. Journal of the academy of marketing science, 48, 137-141.

Reichheld, F. F., & Sasser, W. E., Jr. (1990). Zero defections: Quality comes to services. Harvard Business Review, 68(5), 105–111.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018, April). Anchors: High-precision model-agnostic explanations. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).

Richman, R., & Wüthrich, M. V. (2023). LocalGLMnet: interpretable deep learning for tabular data. Scandinavian Actuarial Journal, 2023(1), 71-95.

Roy, S. K., & Ganguli, S. (2008). Service quality and customer satisfaction: An empirical investigation in Indian mobile telecommunications services. Marketing Management Journal, 18(2), 119–144.

Schwalbe, G., & Finzel, B. (2024). A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. Data Mining and Knowledge Discovery, 38(5), 3043-3101.

Seo, B., & Li, J. (2024). Explainable machine learning by SEE-Net: closing the gap between interpretable models and DNNs. Scientific reports, 14(1), 26302.

Shah, V., & Konda, S. R. (2021). Neural networks and explainable AI: bridging the gap between models and interpretability. International Journal of Computer Science and Technology, 5(2), 163-176.

Shrikumar, A., Greenside, P., & Kundaje, A. (2017, July). Learning important features through propagating activation differences. In International conference on machine learning (pp. 3145-3153). PMlR.

Si, J. Y. H., Cooper, M., Cheng, W. Y., & Krishnan, R. (2023). Interpretabnet: Enhancing interpretability of tabular data using deep generative models and large language models. In NeurIPS 2023 Second Table Representation Learning Workshop.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.

Singh, P. (2021). Leveraging AI for advanced troubleshooting in telecommunications: Enhancing network reliability, customer satisfaction, and social equity. Journal of Science & Technology, 2(2). The Science Brigade Publishers.

Sundararajan, M., Taly, A., & Yan, Q. (2017, July). Axiomatic attribution for deep networks. In International conference on machine learning (pp. 3319-3328). PMLR.

Wachter, S., Mittelstadt, B., & Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. Harv. JL & Tech., 31, 841.

Yang, Z., Zhang, A., & Sudjianto, A. (2021). GAMI-Net: An explainable neural network based on generalized additive models with structured interactions. Pattern Recognition, 120, 108192.

Zhang, W., Barr, B., & Paisley, J. (2024, March). Gaussian process neural additive models. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 15, pp. 16865-16872).

APPENDIX A

Variable	Description
anomaly_rate_box_cpu	Captures the proportion of set-top box CPU related events that are flagged as anomalies, giving insight into unusual set-top box CPU patterns for a specific client. This metric is calculated by comparing each set-top box CPU feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for set-top box CPU is then computed as the total count of these flagged set-top box CPU events divided by the total number of monitored set-top box CPU features.
anomaly_rate_box_tuner	Captures the proportion of set-top box tuner related events that are flagged as anomalies, giving insight into unusual set-top box tuner patterns for a specific client. This metric is calculated by comparing each set-top box tuner feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for set-top box tuner is then computed as the total count of these flagged set-top box tuner events divided by the total number of monitored set-top box tuner features.

anomaly_rate_cpu	Captures the proportion of CPU events that are flagged as anomalies, giving insight into unusual reboot patterns for a specific client. This metric is calculated by comparing each CPU-related feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for CPU is then computed as the total count of these flagged CPU events divided by the total number of monitored CPU-related features.
anomaly_rate_docsis30	Captures the proportion of DOCSIS 3.0 events that are flagged as anomalies, giving insight into unusual DOCSIS 3.0 patterns for a specific client. This metric is calculated by comparing each DOCSIS 3.0-related feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for DOCSIS 3.0 is then computed as the total count of these flagged DOCSIS 3.0 events divided by the total number of monitored DOCSIS 3.0-related features.
anomaly_rate_docsis31	Captures the proportion of DOCSIS 3.1 events that are flagged as anomalies, giving insight into unusual DOCSIS 3.1 patterns for a specific client. This metric is

	calculated by comparing each DOCSIS 3.1-related feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for DOCSIS 3.0 is then computed as the total count of these flagged DOCSIS 3.0 events divided by the total number of monitored DOCSIS 3.0-related features.
anomaly_rate_front_end	Captures the proportion of front-end system related events that are flagged as anomalies, giving insight into unusual front-end system patterns for a specific client. This metric is calculated by comparing each front-end system feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for front-end system is then computed as the total count of these flagged front-end system events divided by the total number of monitored front-end system features.
anomaly_rate_ftth	Captures the proportion of FTTH events that are flagged as anomalies, giving insight into unusual FTTH patterns for a specific client. This metric is calculated by comparing each FTTH-related feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a

	specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for FTTH is then computed as the total count of these flagged FTTH events divided by the total number of monitored FTTH-related features.
anomaly_rate_reboots	Captures the proportion of reboot events that are flagged as anomalies, giving insight into unusual reboot patterns for a specific client. This metric is calculated by comparing each reboot-related feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for reboots is then computed as the total count of these flagged reboot events divided by the total number of monitored reboot-related features.
anomaly_rate_reboots_tv	Captures the proportion of TV related reboot events that are flagged as anomalies, giving insight into unusual TV related reboot patterns for a specific client. This metric is calculated by comparing each TV related reboot feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for TV related reboots is then computed as the total count of

	these flagged TV related reboot events divided by the total number of monitored TV related reboot features.
anomaly_rate_tv_sessions	Captures the proportion of TV sessions events that are flagged as anomalies, giving insight into unusual TV sessions patterns for a specific client. This metric is calculated by comparing each TV sessions feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for TV sessions is then computed as the total count of these flagged TV sessions events divided by the total number of monitored TV sessions features.
anomaly_rate_wifi_issues	Captures the proportion of WIFI events that are flagged as anomalies, giving insight into unusual WIFI patterns for a specific client. This metric is calculated by comparing each WIFI-related feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for WIFI is then computed as the total count of these flagged WIFI events divided by the total number of monitored WIFI-related features.
anomaly_rate_wifi_rssi	Captures the proportion of WIFI-RSSI events that are flagged as

	anomalies, giving insight into unusual WIFI-RSSI patterns for a specific client. This metric is calculated by comparing each WIFI-RSSI related feature against a defined threshold (typically set at the mean + 2 * standard deviations or above a specific percentile). When a feature exceeds this threshold, it is flagged as anomalous. The anomaly rate for WIFI-RSSI is then computed as the total count of these flagged WIFI-RSSI events divided by the total number of monitored WIFI-RSSI related features.
available_day_id	Refers to the day the data became available for use, formatted as YYYYMMDD
c_dwns_bip_err_frames_qos_above_sum_m3d_0d	This variable tracks the number of occurrences, over the last 3 days, where the count of downstream BIP errors (bit-interleaved parity errors) exceeds a certain threshold. BIP errors are detected when a GPON Encapsulation Mode (GEM) frame, used for data transport in GPON systems, is found to be faulty.
c_dwns_bip_err_frames_qos_above_sum_m7d_0d	This variable tracks the number of occurrences, over the last 7 days, where the count of downstream BIP errors (bit-interleaved parity errors) exceeds a certain threshold. BIP errors are detected when a GPON Encapsulation Mode (GEM) frame, used for data transport in

	GPON systems, is found to be faulty.
c_dwns_bip_err_frames_qos_above_sum_m14d_m7d	This variable tracks the number of occurrences, between the 14 and 7 previous days, where the count of downstream BIP errors (bit-interleaved parity errors) exceeds a certain threshold. BIP errors are detected when a GPON Encapsulation Mode (GEM) frame, used for data transport in GPON systems, is found to be faulty.
c_dwns_bip_err_frames_qos_above_sum_m30d_m14d	This variable tracks the number of occurrences, between the 30 and 14 previous days, where the count of downstream BIP errors (bit-interleaved parity errors) exceeds a certain threshold. BIP errors are detected when a GPON Encapsulation Mode (GEM) frame, used for data transport in GPON systems, is found to be faulty.
c_ups_bip_err_frames_qos_above_sum_m3d_0d	This variable tracks the number of occurrences, over the last 3 days, where the count of upstream BIP (Bit-Interleaved Parity) errors exceeds a certain threshold. These errors are detected when a GPON Encapsulation Mode (GEM) frame, used in GPON systems for upstream data transmission, is found to be faulty.
c_ups_bip_err_frames_qos_above_sum_m7d_0d	This variable tracks the number of occurrences, over the last 7 days, where the count of upstream BIP (Bit-Interleaved Parity) errors exceeds a certain threshold. These errors are detected when a GPON

	Encapsulation Mode (GEM) frame, used in GPON systems for upstream data transmission, is found to be faulty.
c_ups_bip_err_frames_qos_above_sum_m14d_m7d	This variable tracks the number of occurrences, between the 14 and 7 previous days, where the count of upstream BIP (Bit-Interleaved Parity) errors exceeds a certain threshold. These errors are detected when a GPON Encapsulation Mode (GEM) frame, used in GPON systems for upstream data transmission, is found to be faulty.
c_ups_bip_err_frames_qos_above_sum_m30d_m14d	This variable tracks the number of occurrences, between the 30 and 14 previous days, where the count of upstream BIP (Bit-Interleaved Parity) errors exceeds a certain threshold. These errors are detected when a GPON Encapsulation Mode (GEM) frame, used in GPON systems for upstream data transmission, is found to be faulty.
c_ups_frame_fec_dec_cor_qos_mean_m3d_0d	This variable represents the average number of upstream Frame Error Corrections (FEC) during the last 3 days. FEC is a process used to detect and correct errors in data transmission.
c_ups_frame_fec_dec_cor_qos_mean_m7d_0d	This variable represents the average number of upstream Frame Error Corrections (FEC) during the last 7 days. FEC is a process used to detect and correct errors in data transmission.

c_ups_frame_fec_dec_cor_qos_mean_m14d_m7d	This variable represents the average number of upstream Frame Error Corrections (FEC) during the period between 14 and 7 days ago. FEC is a process used to detect and correct errors in data transmission.
c_ups_frame_fec_dec_cor_qos_mean_m30d_m14d	This variable represents the average number of upstream Frame Error Corrections (FEC) during the period between 30 and 14 days ago. FEC is a process used to detect and correct errors in data transmission.
c_ups_frame_fec_dec_n_cor_qos_mean_m3d_0d	This variable represents the average number of upstream frame errors that could not be corrected by the Forward Error Correction (FEC) mechanism in the last 3 days. These uncorrected errors indicate severe transmission issues, where data corruption could not be resolved by FEC.
c_ups_frame_fec_dec_n_cor_qos_mean_m7d_0d	This variable represents the average number of upstream frame errors that could not be corrected by the Forward Error Correction (FEC) mechanism in the last 7 days. These uncorrected errors indicate severe transmission issues, where data corruption could not be resolved by FEC.
c_ups_frame_fec_dec_n_cor_qos_mean_m14d_m7d	This variable represents the average number of upstream frame errors that could not be corrected by the Forward Error Correction (FEC) mechanism between 14 and 7 days ago. These uncorrected errors

	indicate severe transmission issues, where data corruption could not be resolved by FEC.
c_ups_frame_fec_dec_n_cor_qos_mean_m30d_m14d	This variable represents the average number of upstream frame errors that could not be corrected by the Forward Error Correction (FEC) mechanism between 30 and 14 days ago. These uncorrected errors indicate severe transmission issues, where data corruption could not be resolved by FEC.
c_ups_scape_err_frames_qos_mean_m3d_0d	Tracks the mean number of upstream error frames in the last 3 days. The errors seem to be related to data frames that encounter issues during upstream transmission (upload). The variable compares the quality of service (QoS) based on these errors over the two periods, helping to identify if the quality of upstream data transmission has improved or worsened over time. Though the exact meaning of "scape" isn't clear, the variable seems to focus on tracking errors and their effect on QoS in upstream communication.
c_ups_scape_err_frames_qos_mean_m7d_0d	Tracks the mean number of upstream error frames in the last 7 days. The errors seem to be related to data frames that encounter issues during upstream transmission (upload). The variable compares the quality of service (QoS) based on these errors over the two periods, helping to identify if the quality of upstream data

	transmission has improved or worsened over time. Though the exact meaning of "scape" isn't clear, the variable seems to focus on tracking errors and their effect on QoS in upstream communication.
c_ups_scape_err_frames_qos_mean_m14d_m7d	Tracks the mean number of upstream error frames between the 14 and 7 previous days. The errors seem to be related to data frames that encounter issues during upstream transmission (upload). The variable compares the quality of service (QoS) based on these errors over the two periods, helping to identify if the quality of upstream data transmission has improved or worsened over time. Though the exact meaning of "scape" isn't clear, the variable seems to focus on tracking errors and their effect on QoS in upstream communication.
c_ups_scape_err_frames_qos_mean_m30d_m14d	Tracks the mean number of upstream error frames between the 30 and 14 previous days. The errors seem to be related to data frames that encounter issues during upstream transmission (upload). The variable compares the quality of service (QoS) based on these errors over the two periods, helping to identify if the quality of upstream data transmission has improved or worsened over time. Though the exact meaning of "scape" isn't clear, the variable seems to focus

	on tracking errors and their effect on QoS in upstream communication.
channelhops24_countDistinct_m3d_0d	Represents the count of distinct Wi-Fi channel changes that occurred on the 2.4 GHz frequency band over the last 3 days. This metric helps to understand how frequently the Wi-Fi system had to switch channels to avoid interference or improve signal quality.
channelhops24_countDistinct_m7d_0d	Represents the count of distinct Wi-Fi channel changes that occurred on the 2.4 GHz frequency band over the last 7 days. This metric helps to understand how frequently the Wi-Fi system had to switch channels to avoid interference or improve signal quality.
channelhops24_countDistinct_m14d_m7d	Represents the count of distinct Wi-Fi channel changes that occurred on the 2.4 GHz frequency band between the 14 and 7 previous days. This metric helps to understand how frequently the Wi-Fi system had to switch channels to avoid interference or improve signal quality.
channelhops24_countDistinct_m30d_m14d	Represents the count of distinct Wi-Fi channel changes that occurred on the 2.4 GHz frequency band between the 30 and 14 previous days. This metric helps to understand how frequently the Wi-Fi system had to switch channels to

	avoid interference or improve signal quality.
channelhops5_countDistinct_m3d_0d	Represents the count of distinct Wi-Fi channel changes that occurred on the 5 GHz frequency band over the last 3 days. This metric helps to understand how frequently the Wi-Fi system had to switch channels to avoid interference or improve signal quality.
channelhops5_countDistinct_m7d_0d	Represents the count of distinct Wi-Fi channel changes that occurred on the 5 GHz frequency band over the last 7 days. This metric helps to understand how frequently the Wi-Fi system had to switch channels to avoid interference or improve signal quality.
channelhops5_countDistinct_m14d_m7d	Represents the count of distinct Wi-Fi channel changes that occurred on the 5 GHz frequency band between the 14 and 7 previous days. This metric helps to understand how frequently the Wi-Fi system had to switch channels to avoid interference or improve signal quality.
channelhops5_countDistinct_m30d_m14d	Represents the count of distinct Wi-Fi channel changes that occurred on the 5 GHz frequency band between the 30 and 14 previous days. This metric helps to understand how frequently the Wi-Fi system had to switch channels to avoid interference or improve signal quality.

complaints_1y	Count of complaints in the last 1 year
count_equipments_net	The count of network-related equipment that the client has
count_equipments_net_anomaly	The count of network-related equipment that the client has which are flagged with anomalies (i.e., equipment experiencing issues or abnormal behavior).
count_equipments_tv	The count of TV-related equipment that the client has
count_equipments_tv_anomaly	The count of TV-related equipment that the client has which are flagged with anomalies (i.e., equipment experiencing issues or abnormal behavior).
cpu_usage_median_m3d_0d	Median cpu usage in the last 3 days. V4s have cpu usage percentage, to get cpu usage multiply by 100
cpu_usage_median_m7d_0d	Median cpu usage in the last 7 days. V4s have cpu usage percentage, to get cpu usage multiply by 100
cpu_usage_median_m14d_m7d	Median cpu usage between the 14 and 7 previous days. V4s have cpu usage percentage, to get cpu usage multiply by 100
cpu_usage_median_m30d_m14d	Median cpu usage between the 30 and 14 previous days. V4s have cpu usage percentage, to get cpu usage multiply by 100
down_disc_frame_qos_mean_m3d_0d	This variable represents the average number of discarded downstream frames in the last 3 days. These are frames that were sent from the ONT to the CMTS but

	were dropped or discarded due to transmission issues, such as errors or congestion.
down_disc_frame_qos_mean_m7d_0d	This variable represents the average number of discarded downstream frames in the last 7 days. These are frames that were sent from the ONT to the CMTS but were dropped or discarded due to transmission issues, such as errors or congestion.
down_disc_frame_qos_mean_m14d_m7d	This variable represents the average number of discarded downstream frames between 14 and 7 days ago. These are frames that were sent from the ONT to the CMTS but were dropped or discarded due to transmission issues, such as errors or congestion.
down_disc_frame_qos_mean_m30d_m14d	This variable represents the average number of discarded downstream frames between 30 and 14 days ago. These are frames that were sent from the ONT to the CMTS but were dropped or discarded due to transmission issues, such as errors or congestion.
down_frame_median_m3d_0d	Median down_frame in the last 3 days. This will be the median usage per hour
down_frame_median_m7d_0d	Median down_frame in the last 7 days. This will be the median usage per hour

down_frame_median_diff_m3d_0d_m7d_0d	Difference of down frame between the last 3 days and last 7 days time periods
down_frame_median_diff_m3d_0d_m30d_m14d	Difference of down frame between the last 3 days and the 30 to 14 previous days time periods
down_frame_median_diff_m3d_0d_m60d_m30d	Difference of down frame between the last 3 days and the 60 to 30 previous days time periods
down_frame_median_diff_m7d_0d_m30d_m14d	Difference of down frame between the last 7 days and the 30 to 14 previous days time periods
down_frame_median_diff_m7d_0d_m60d_m30d	Difference of down frame between the last 7 days and the 60 to 30 previous days time periods
down_frame_median_m14d_m7d	Median down_frame between the 14 and 7 previous days. This will be the median usage per hour
down_frame_median_m30d_m14d	Median down_frame between the 30 and 14 previous days. This will be the median usage per hour
down_frame_median_m60d_m30d	Median down_frame between the 60 and 30 previous days. This will be the median usage per hour
dwn_avg_throughput_mean_m3d_0d	This variable represents the average downstream throughput (in Kbps) measured over the last 3 days. It calculates the mean data transmission rate from the ONT to the OLT in the upstream direction during this time frame.
dwn_avg_throughput_mean_m7d_0d	This variable represents the average downstream throughput (in Kbps) measured over the last 7 days. It calculates the mean data

	transmission rate from the ONT to the OLT in the upstream direction during this time frame.
dwn_avg_throughput_mean_m14d_m7d	This variable represents the average downstream throughput (in Kbps) measured over the period from 14 to 7 days ago. It calculates the mean data transmission rate from the ONT to the OLT in the upstream direction during this time frame.
dwn_avg_throughput_mean_m30d_m14d	This variable represents the average downstream throughput (in Kbps) measured over the period from 30 to 14 days ago. It calculates the mean data transmission rate from the ONT to the OLT in the upstream direction during this time frame.
dwn_avg_throughput_stddev_m3d_0d	This variable represents the standard deviation of the downstream throughput (in Kbps) in the last 3 days. It measures the variability or fluctuation in the upstream data transmission rate during that time period.
dwn_avg_throughput_stddev_m7d_0d	This variable represents the standard deviation of the downstream throughput (in Kbps) in the last 7 days. It measures the variability or fluctuation in the upstream data transmission rate during that time period.
dwn_avg_throughput_stddev_m14d_m7d	This variable represents the standard deviation of the downstream throughput (in Kbps) from 14 to 7 days ago. It measures the variability or fluctuation in the

	upstream data transmission rate during that time period.
dwn_avg_throughput_stddev_m30d_m14d	This variable represents the standard deviation of the downstream throughput (in Kbps) from 30 to 14 days ago. It measures the variability or fluctuation in the upstream data transmission rate during that time period.
flg_ftth	A flag indicating whether the client has a Fiber to the Home (FTTH) connection
free_mem_perc_median_m3d_0d	Median freememperc in the last 3 days. freememperc = memory_free/memory_total
free_mem_perc_median_m7d_0d	Median freememperc in the last 7 days. freememperc = memory_free/memory_total
free_mem_perc_median_m14d_m7d	Median freememperc between the 14 and 7 previous days. freememperc = memory_free/memory_total
free_mem_perc_median_m30d_m14d	Median freememperc between the 30 and 14 previous days. freememperc = memory_free/memory_total
function_fe_count_m3d_0d	Total functions used in the last 3 days
function_fe_count_m6d_0d	Total functions used in the last 6 days
function_fe_count_m14d_m6d	Total functions used between the 14 and 6 previous days
function_fe_count_m30d_m14d	Total functions used between the 30 and 14 previous days

high_uptime_sum_m14d_0d	Sum of times where uptime was above 540434 (99% percentile) in the last 14 days
high_uptime_sum_m30d_0d	Sum of times where uptime was above 540434 (99% percentile) in the last 30 days
high_uptime_sum_m3d_0d	Sum of times where uptime was above 540434 (99% percentile) in the last 3 days
high_uptime_sum_m7d_0d	Sum of times where uptime was above 540434 (99% percentile) in the last 7 days
httprequest_csbytes_sum_m3d_0d	Sum of bytes of the request in the last 3 days(Only available for IRISs)
httprequest_csbytes_sum_m6d_0d	Sum of bytes of the request in the last 6 days (Only available for IRISs)
httprequest_csbytes_sum_m14d_m6d	Sum of bytes of the request between the 14 and 6 previous days (Only available for IRISs)
httprequest_csbytes_sum_m30d_m14d	Sum of bytes of the request between the 30 and 14 previous days (Only available for IRISs)
httpresponse_scbytes_sum_m3d_0d	Sum of bytes of the response in the last 3 days(Only available for IRISs)
httpresponse_scbytes_sum_m6d_0d	Sum of bytes of the response in the last 6 days (Only available for IRISs)
httpresponse_scbytes_sum_m14d_m6d	Sum of bytes of the response between the 14 and 6 previous days (Only available for IRISs)
httpresponse_scbytes_sum_m30d_m14d	Sum of bytes of the response between the 30 and 14 previous days (Only available for IRISs)

hw_mem_perc_median_m3d_0d	Median hardware mem_perc (mem_perc = memory used/memory free) in the last 3 days (Not available for v4, v5 hitron and v5 sagem)
hw_mem_perc_median_m7d_0d	Median hardware mem_perc (mem_perc = memory used/memory free) in the last 7 days (Not available for v4, v5 hitron and v5 sagem)
hw_mem_perc_median_m14d_m7d	Median hardware mem_perc (mem_perc = memory used/memory free) between the 14 and 7 previous days (Not available for v4, v5 hitron and v5 sagem)
hw_mem_perc_median_m30d_m14d	Median hardware mem_perc (mem_perc = memory used/memory free) between the 30 and 14 previous days (Not available for v4, v5 hitron and v5 sagem)
interactions_1y	Count of interactions in the last 1 year
label	Customers with or without an OT (Ordem de Trabalho) scheduled in the next 7 days
low_uptime_sum_m14d_0d	Sum of times where uptime was below 1454 (1% percentile) in the last 14 days
low_uptime_sum_m30d_0d	Sum of times where uptime was below 1454 (1% percentile) in the last 30 days

low_uptime_sum_m3d_0d	Sum of times where uptime was below 1454 (1% percentile) in the last 3 days
low_uptime_sum_m7d_0d	Sum of times where uptime was below 1454 (1% percentile) in the last 7 days
model_segment_id_net	The model ID of the client's network equipment (the router)
model_segment_id_tv	The model ID of the client's TV equipment (the box)
no_bytes_down_sum_m3d_0d	Count of times per hour over the last 3 days where no bytes were downloaded. Essentially, it tracks how often, on an hourly basis, there was no download activity in the past week.
no_bytes_down_sum_m7d_0d	Count of times per hour over the last 7 days where no bytes were downloaded. Essentially, it tracks how often, on an hourly basis, there was no download activity in the past week.
no_bytes_down_sum_m14d_m7d	Count of times per hour between the 14 and 7 previous days where no bytes were downloaded. Essentially, it tracks how often, on an hourly basis, there was no download activity in the past week.
no_bytes_down_sum_m30d_m14d	Count of times per hour between the 30 and 14 previous days where no bytes were downloaded. Essentially, it tracks how often, on an hourly basis, there was no download activity in the past week.
no_bytes_down_sum_m60d_m30d	Count of times per hour between the 60 and 30 previous days where

	no bytes were downloaded. Essentially, it tracks how often, on an hourly basis, there was no download activity in the past week.
no_bytes_up_sum_m3d_0d	Count of times per hour over the last 3 days where no bytes were uploaded. Essentially, it tracks how often, on an hourly basis, there was no upload activity in the past week.
no_bytes_up_sum_m7d_0d	Count of times per hour over the last 7 days where no bytes were uploaded. Essentially, it tracks how often, on an hourly basis, there was no upload activity in the past week.
no_bytes_up_sum_m14d_m7d	Count of times per hour between the 14 and 7 previous days where no bytes were uploaded. Essentially, it tracks how often, on an hourly basis, there was no upload activity in the past week.
no_bytes_up_sum_m30d_m14d	Count of times per hour between the 30 and 14 previous days where no bytes were uploaded. Essentially, it tracks how often, on an hourly basis, there was no upload activity in the past week.
no_bytes_up_sum_m60d_m30d	Count of times per hour between the 60 and 30 previous days where no bytes were uploaded. Essentially, it tracks how often, on an hourly basis, there was no upload activity in the past week.
no_down_frame_sum_m3d_0d	Flag that counts the number of days with no down_frames, i.e., down_frames = 0, in the last 3 days

no_down_frame_sum_m7d_0d	Flag that counts the number of days with no down_frames, i.e., down_frames = 0, in the last 7 days
no_down_frame_sum_m14d_m7d	Flag that counts the number of days with no down_frames, i.e., down_frames = 0, between the 14 and 7 previous days
no_down_frame_sum_m30d_m14d	Flag that counts the number of days with no down_frames, i.e., down_frames = 0, between the 30 and 14 previous days
no_up_frame_sum_m3d_0d	Flag that counts the number of days with no up_frames, i.e., up_frames = 0, in the last 3 days
no_up_frame_sum_m7d_0d	Flag that counts the number of days with no up_frames, i.e., up_frames = 0, in the last 7 days
no_up_frame_sum_m14d_m7d	Flag that counts the number of days with no up_frames, i.e., up_frames = 0, between the 14 and 7 previous days
no_up_frame_sum_m30d_m14d	Flag that counts the number of days with no up_frames, i.e., up_frames = 0, between the 30 and 14 previous days
npvr_session_30s_sum_m3d_0d	Count of times there were sessions with less than 30 seconds (short sessions) in the last 3 days, using a NPVR session
npvr_session_30s_sum_m7d_0d	Count of times there were sessions with less than 30 seconds (short sessions) in the last 7 days, using a NPVR session
npvr_session_30s_sum_m14d_m7d	Count of times there were sessions with less than 30 seconds (short

	sessions) between the 14 and 7 previous days, using a NPVR session
npvr_session_30s_sum_m30d_m14d	Count of times there were sessions with less than 30 seconds (short sessions) between the 30 and 14 previous days, using a NPVR session
num_processes_median_m3d_0d	Median number of process in the last 3 days (Not available for v4s)
num_processes_median_m7d_0d	Median number of process in the last 7 days (Not available for v4s)
num_processes_median_m14d_m7d	Median number of process between the 14 and 7 previous days (Not available for v4s)
num_processes_median_m30d_m14d	Median number of process between the 30 and 14 previous days (Not available for v4s)
numdevs24_mean_m3d_0d	Represents the average number of devices connected to the 2.4 GHz frequency band over the last 3 days. It gives the mean count of connected devices during this period.
numdevs24_mean_m7d_0d	Represents the average number of devices connected to the 2.4 GHz frequency band over the last 7 days. It gives the mean count of connected devices during this period.
numdevs24_mean_m14d_m7d	Represents the average number of devices connected to the 2.4 GHz frequency band between the 14 and 7 previous days. It gives the mean count of connected devices during this period.

numdevs24_mean_m30d_m14d	Represents the average number of devices connected to the 2.4 GHz frequency band between the 30 and 14 previous days. It gives the mean count of connected devices during this period.
numdevs24_stddev_m3d_0d	Represents the standard deviation of the number of devices connected on the 2.4 GHz frequency band over the last 3 days. It shows how much the number of connected devices fluctuates within this time period.
numdevs24_stddev_m7d_0d	Represents the standard deviation of the number of devices connected on the 2.4 GHz frequency band over the last 7 days. It shows how much the number of connected devices fluctuates within this time period.
numdevs24_stddev_m14d_m7d	Represents the standard deviation of the number of devices connected on the 2.4 GHz frequency band between the 14 and 7 previous days. It shows how much the number of connected devices fluctuates within this time period.
numdevs24_stddev_m30d_m14d	Represents the standard deviation of the number of devices connected on the 2.4 GHz frequency band between the 30 and 14 previous days. It shows how much the number of connected devices fluctuates within this time period.
numdevs5_mean_m3d_0d	Represents the average number of devices connected to the 5 GHz

	frequency band over the last 3 days. It gives the mean count of connected devices during this period.
numdevs5_mean_m7d_0d	Represents the average number of devices connected to the 5 GHz frequency band over the last 7 days. It gives the mean count of connected devices during this period.
numdevs5_mean_m14d_m7d	Represents the average number of devices connected to the 5 GHz frequency band between the 14 and 7 previous days. It gives the mean count of connected devices during this period.
numdevs5_mean_m30d_m14d	Represents the average number of devices connected to the 5 GHz frequency band between the 30 and 14 previous days. It gives the mean count of connected devices during this period.
numdevs5_stddev_m3d_0d	Represents the standard deviation of the number of devices connected on the 5 GHz frequency band over the last 3 days. It shows how much the number of connected devices fluctuates within this time period.
numdevs5_stddev_m7d_0d	Represents the standard deviation of the number of devices connected on the 5 GHz frequency band over the last 7 days. It shows how much the number of connected devices fluctuates within this time period.

numdevs5_stddev_m14d_m7d	Represents the standard deviation of the number of devices connected on the 5 GHz frequency band between the 14 and 7 previous days. It shows how much the number of connected devices fluctuates within this time period.
numdevs5_stddev_m30d_m14d	Represents the standard deviation of the number of devices connected on the 5 GHz frequency band between the 30 and 14 previous days. It shows how much the number of connected devices fluctuates within this time period.
ont_tx_power_select_1310_above_sum_m3d_0d	This variable tracks the number of instances, over the last 3 days, where the ONT transmission power (TX Power) for the return path to NET and VOD services (at a wavelength of 1310 nm) exceeds a defined threshold.
ont_tx_power_select_1310_above_sum_m7d_0d	This variable tracks the number of instances, over the last 7 days, where the ONT transmission power (TX Power) for the return path to NET and VOD services (at a wavelength of 1310 nm) exceeds a defined threshold.
ont_tx_power_select_1310_above_sum_m14d_m7d	This variable tracks the number of instances, between the 14 and 7 previous days, where the ONT transmission power (TX Power) for the return path to NET and VOD services (at a wavelength of 1310 nm) exceeds a defined threshold.
ont_tx_power_select_1310_above_sum_m30d_m14d	This variable tracks the number of instances, between the 30 and 14 previous days, where the ONT

	transmission power (TX Power) for the return path to NET and VOD services (at a wavelength of 1310 nm) exceeds a defined threshold.
ots_1y	Count of OTs in the last 1 year
pts_1y	Count of PTs in the last 1 year
routerstate_notready_sum_m3d_0d	Refers to the sum of times (or instances) over the last 3 days when the router was not in the "READY" state. The "READY" state typically means the router is fully functional and operational, so this metric counts how often it was in any other state within the last 3 days.
routerstate_notready_sum_m7d_0d	Refers to the sum of times (or instances) over the last 7 days when the router was not in the "READY" state. The "READY" state typically means the router is fully functional and operational, so this metric counts how often it was in any other state within the last 7 days.
routerstate_notready_sum_m14d_m7d	Refers to the sum of times (or instances) between the 14 and 7 previous days when the router was not in the "READY" state. The "READY" state typically means the router is fully functional and operational, so this metric counts how often it was in any other state within the last 7 days.
routerstate_notready_sum_m30d_m14d	Refers to the sum of times (or instances) between the 30 and 14 previous days when the router was not in the "READY" state. The

	"READY" state typically means the router is fully functional and operational, so this metric counts how often it was in any other state within the last 7 days.
session_id_count_m3d_0d	Count of sessions in the last 3 days
session_id_count_m7d_0d	Count of sessions in the last 7 days
session_id_count_m14d_m7d	Count of sessions between the 14 and 7 previous days
session_id_count_m30d_m14d	Count of sessions between the 30 and 14 previous days
session_time_sec_mean_m3d_0d	Average duration of the sessions in the last 3 days
session_time_sec_mean_m7d_0d	Average duration of the sessions in the last 7 days
session_time_sec_mean_m14d_m7d	Average duration of the sessions between the 14 and 7 previous days
session_time_sec_mean_m30d_m14d	Average duration of the sessions between the 30 and 14 previous days
short_term_anomaly_rate_box_cpu	Represents the proportion of set-top box CPU-related events flagged as anomalies for a client over the short-term period (3 and 7 days).
short_term_anomaly_rate_box_tuner	Represents the proportion of set-top box tuner-related events flagged as anomalies for a client over the short-term period (3 and 7 days).
short_term_anomaly_rate_cpu	Represents the proportion of CPU-related events flagged as anomalies

	for a client over the short-term period (3 and 7 days).
short_term_anomaly_rate_docsis30	Represents the proportion of DOCSIS 3.0 events flagged as anomalies for a client over the short-term period (3 and 7 days).
short_term_anomaly_rate_docsis31	Represents the proportion of DOCSIS 3.1 events flagged as anomalies for a client over the short-term period (3 and 7 days).
short_term_anomaly_rate_front_end	Represents the proportion of front-end system events flagged as anomalies for a client over the short-term period (3 and 7 days).
short_term_anomaly_rate_ftth	Represents the proportion of FTTH (Fiber to the Home) events flagged as anomalies for a client over the short-term period (3 and 7 days).
short_term_anomaly_rate_reboots	Represents the proportion of reboot events flagged as anomalies for a client over the short-term period (3 and 7 days).
short_term_anomaly_rate_reboots_tv	Represents the proportion of TV-related reboot events flagged as anomalies for a client over the short-term period (3 and 7 days).
short_term_anomaly_rate_tv_sessions	Represents the proportion of TV session events flagged as anomalies for a client over the short-term period (3 and 7 days).
short_term_anomaly_rate_wifi_issues	Represents the proportion of Wi-Fi-related issues flagged as anomalies for a client over the short-term period (3 and 7 days).

short_term_anomaly_rate_wifi_rssi	Represents the proportion of Wi-Fi RSSI (signal strength) events flagged as anomalies for a client over the short-term period (3 and 7 days).
sum_bytes_down_median_m3d_0d	Median total bytes downloaded in the last 3 days
sum_bytes_down_median_m7d_0d	Median total bytes downloaded in the last 7 days
sum_bytes_down_median_diff_m3d_0d_m7d_0d	Difference of median total bytes downloaded between the last 3 days and last 7 days time periods
sum_bytes_down_median_diff_m3d_0d_m30d_m14d	Difference of median total bytes downloaded between the last 3 days and the 30 to 14 previous days time periods
sum_bytes_down_median_diff_m3d_0d_m60d_m30d	Difference of median total bytes downloaded between the last 3 days and the 60 to 30 previous days time periods
sum_bytes_down_median_diff_m7d_0d_m30d_m14d	Difference of median total bytes downloaded between the last 7 days and the 30 to 14 previous days time periods
sum_bytes_down_median_diff_m7d_0d_m60d_m30d	Difference of median total bytes downloaded between the last 7 days and the 60 to 30 previous days time periods
sum_bytes_down_median_m14d_m7d	Median total bytes downloaded between the 14 and 7 previous days
sum_bytes_down_median_m30d_m14d	Median total bytes downloaded between the 30 and 14 previous days

sum_bytes_down_median_m60d_m30d	Median total bytes downloaded between the 60 and 30 previous days
sum_bytes_up_median_m3d_0d	Median total bytes uploaded in the last 3 days
sum_bytes_up_median_m7d_0d	Median total bytes uploaded in the last 7 days
sum_bytes_up_median_diff_m3d_0d_m7d_0d	Difference of median total bytes uplodaded between the last 3 days and last 7 days time periods
sum_bytes_up_median_diff_m3d_0d_m30d_m14d	Difference of median total bytes uplodaded between the last 3 days and the 30 to 14 previous days time periods
sum_bytes_up_median_diff_m3d_0d_m60d_m30d	Difference of median total bytes uplodaded between the last 3 days and the 60 to 30 previous days time periods
sum_bytes_up_median_diff_m7d_0d_m30d_m14d	Difference of median total bytes uplodaded between the last 7 days and the 30 to 14 previous days time periods
sum_bytes_up_median_diff_m7d_0d_m60d_m30d	Difference of median total bytes uplodaded between the last 7 days and the 60 to 30 previous days time periods
sum_bytes_up_median_m14d_m7d	Median total bytes uploaded between the 14 and 7 previous days
sum_bytes_up_median_m30d_m14d	Median total bytes uploaded between the 30 and 14 previous days

sum_bytes_up_median_m60d_m30d	Median total bytes uploaded between the 60 and 30 previous days
temp_2gchip_cpu_median_m3d_0d	Median 2.4g chip temperature in the last 3 days (Not available for v4, v5 hitron and v5 sagem)
temp_2gchip_cpu_median_m7d_0d	Median 2.4g chip temperature in the last 7 days (Not available for v4, v5 hitron and v5 sagem)
temp_2gchip_cpu_median_m14d_m7d	Median 2.4g chip temperature between the 14 and 7 previous days (Not available for v4, v5 hitron and v5 sagem)
temp_2gchip_cpu_median_m30d_m14d	Median 2.4g chip temperature between the 30 and 14 previous days (Not available for v4, v5 hitron and v5 sagem)
temp_5gchip_cpu_median_m3d_0d	Median 5g chip temperature in the last 3 days (Not available for v4, v5 hitron and v5 sagem)
temp_5gchip_cpu_median_m7d_0d	Median 5g chip temperature in the last 7 days (Not available for v4, v5 hitron and v5 sagem)
temp_5gchip_cpu_median_m14d_m7d	Median 5g chip temperature between the 14 and 7 previous days (Not available for v4, v5 hitron and v5 sagem)
temp_5gchip_cpu_median_m30d_m14d	Median 5g chip temperature between the 30 and 14 previous days (Not available for v4, v5 hitron and v5 sagem)
temp_main_cpu_median_m3d_0d	Median main chip temperature in the last 3 days (Not available for v4, v5 hitron and v5 sagem)

temp_main_cpu_median_m7d_0d	Median main chip temperature in the last 7 days (Not available for v4, v5 hitron and v5 sagem)
temp_main_cpu_median_m14d_m7d	Median main chip temperature between the 14 and 7 previous days (Not available for v4, v5 hitron and v5 sagem)
temp_main_cpu_median_m30d_m14d	Median main chip temperature between the 30 and 14 previous days (Not available for v4, v5 hitron and v5 sagem)
temp_select_ont_above_sum_m3d_0d	This variable tracks the number of occurrences, over the last 3 days, where the ONT temperature exceeded a certain threshold. When the ONT operates at temperatures above this level, it may stop functioning or experience performance issues.
temp_select_ont_above_sum_m7d_0d	This variable tracks the number of occurrences, over the last 7 days, where the ONT temperature exceeded a certain threshold. When the ONT operates at temperatures above this level, it may stop functioning or experience performance issues.
temp_select_ont_above_sum_m14d_m7d	This variable tracks the number of occurrences, between the 14 and 7 previous days, where the ONT temperature exceeded a certain threshold. When the ONT operates at temperatures above this level, it may stop functioning or experience performance issues.
temp_select_ont_above_sum_m30d_m14d	This variable tracks the number of occurrences, between the 30 and

	14 previous days, where the ONT temperature exceeded a certain threshold. When the ONT operates at temperatures above this level, it may stop functioning or experience performance issues.
up_avg_throughput_mean_m3d_0d	This variable represents the average upstream throughput (in Kbps) measured over the last 3 days. It calculates the mean data transmission rate from the ONT to the OLT in the upstream direction during this time frame.
up_avg_throughput_mean_m7d_0d	This variable represents the average upstream throughput (in Kbps) measured over the last 3 days. It calculates the mean data transmission rate from the ONT to the OLT in the upstream direction during this time frame.
up_avg_throughput_mean_m14d_m7d	This variable represents the average upstream throughput (in Kbps) measured over the period from 14 to 7 days ago. It calculates the mean data transmission rate from the ONT to the OLT in the upstream direction during this time frame.
up_avg_throughput_mean_m30d_m14d	This variable represents the average upstream throughput (in Kbps) measured over the period from 30 to 14 days ago. It calculates the mean data transmission rate from the ONT to the OLT in the upstream direction during this time frame.
up_avg_throughput_stddev_m3d_0d	This variable represents the standard deviation of the upstream

	throughput (in Kbps) in the last 3 days. It measures the variability or fluctuation in the upstream data transmission rate during that time period.
up_avg_throughput_stddev_m7d_0d	This variable represents the standard deviation of the upstream throughput (in Kbps) in the last 7 days. It measures the variability or fluctuation in the upstream data transmission rate during that time period.
up_avg_throughput_stddev_m14d_m7d	This variable represents the standard deviation of the upstream throughput (in Kbps) from 14 to 7 days ago. It measures the variability or fluctuation in the upstream data transmission rate during that time period.
up_avg_throughput_stddev_m30d_m14d	This variable represents the standard deviation of the upstream throughput (in Kbps) from 30 to 14 days ago. It measures the variability or fluctuation in the upstream data transmission rate during that time period.
up_disc_frame_qos_mean_m3d_0d	This variable represents the average number of discarded upstream frames in the last 3 days. These are frames that were sent from the ONT to the CMTS but were dropped or discarded due to transmission issues, such as errors or congestion.
up_disc_frame_qos_mean_m7d_0d	This variable represents the average number of discarded upstream frames in the last 7 days. These are frames that were sent

	from the ONT to the CMTS but were dropped or discarded due to transmission issues, such as errors or congestion.
up_disc_frame_qos_mean_m14d_m7d	This variable represents the average number of discarded upstream frames between 14 and 7 days ago. These are frames that were sent from the ONT to the CMTS but were dropped or discarded due to transmission issues, such as errors or congestion.
up_disc_frame_qos_mean_m30d_m14d	This variable represents the average number of discarded upstream frames between 30 and 14 days ago. These are frames that were sent from the ONT to the CMTS but were dropped or discarded due to transmission issues, such as errors or congestion.
up_frame_median_m3d_0d	Median up_frame in the last 3 days. This will be the median usage per hour
up_frame_median_m7d_0d	Median up_frame in the last 7 days. This will be the median usage per hour
up_frame_median_diff_m3d_0d_m7d_0d	Difference of up frame between the last 3 days and last 7 days time periods
up_frame_median_diff_m3d_0d_m30d_m14d	Difference of up frame between the last 3 days and the 30 to 14 previous days time periods

up_frame_median_diff_m3d_0d_m60d_m30d	Difference of up frame between the last 3 days and the 60 to 30 previous days time periods
up_frame_median_diff_m7d_0d_m30d_m14d	Difference of up frame between the last 7 days and the 30 to 14 previous days time periods
up_frame_median_diff_m7d_0d_m60d_m30d	Difference of up frame between the last 7 days and the 60 to 30 previous days time periods
up_frame_median_m14d_m7d	Median up_frame between the 14 and 7 previous days. This will be the median usage per hour
up_frame_median_m30d_m14d	Median up_frame between the 30 and 14 previous days. This will be the median usage per hour
up_frame_median_m60d_m30d	Median up_frame between the 60 and 30 previous days. This will be the median usage per hour
up_time_median_m3d_0d	Get the raw uptime and divide by then 60 then 60 to get the uptime in hours for V5s, for V4s same thing but do an extra division by 1000. Median of uptime in the last 3 days. Uptime is the total time that the equipment is on, so, if a customer has their equipment on for the whole week they would have uptime = 7 days * 24 hours = 168, if it keeps turned on for one more week then the uptime would be 168 + 168 = 336
up_time_median_m7d_0d	Get the raw uptime and divide by then 60 then 60 to get the uptime in hours for V5s, for V4s same thing but do an extra division by 1000. Median of uptime in the last 7 days. Uptime is the total time that the

	equipment is on, so, if a customer has their equipment on for the whole week they would have uptime = 7 days * 24 hours = 168, if it keeps turned on for one more week then the uptime would be 168 + 168 = 336
up_time_median_m14d_m7d	Get the raw uptime and divide by then 60 then 60 to get the uptime in hours for V5s, for V4s same thing but do an extra division by 1000. Median of uptime between the 14 and 7 previous days. Uptime is the total time that the equipment is on, so, if a customer has their equipment on for the whole week they would have uptime = 7 days * 24 hours = 168, if it keeps turned on for one more week then the uptime would be 168 + 168 = 336
up_time_median_m30d_m14d	Get the raw uptime and divide by then 60 then 60 to get the uptime in hours for V5s, for V4s same thing but do an extra division by 1000. Median of uptime between the 30 and 14 previous days. Uptime is the total time that the equipment is on, so, if a customer has their equipment on for the whole week they would have uptime = 7 days * 24 hours = 168, if it keeps turned on for one more week then the uptime would be 168 + 168 = 336
upgrades_countDistinct_m3d_0d	This variable tracks the distinct count of software upgrades for the router over the last 3 days

upgrades_countDistinct_m7d_0d	This variable tracks the distinct count of software upgrades for the router over the last 7 days
upgrades_countDistinct_m14d_m7d	This variable tracks the distinct count of software upgrades for the router between the 14 and 7 previous days
upgrades_countDistinct_m30d_m14d	This variable tracks the distinct count of software upgrades for the router between the 30 and 14 previous days
vod_session_30s_sum_m3d_0d	Count of times there were sessions with less than 30 seconds (short sessions) in the last 3 days, using a VOD session
vod_session_30s_sum_m7d_0d	Count of times there were sessions with less than 30 seconds (short sessions) in the last 7 days, using a VOD session
vod_session_30s_sum_m14d_m7d	Count of times there were sessions with less than 30 seconds (short sessions) between the 14 and 7 previous days, using a VOD session
vod_session_30s_sum_m30d_m14d	Count of times there were sessions with less than 30 seconds (short sessions) between the 30 and 14 previous days, using a VOD session
with_anomaly_1y	Count of OTs with anomaly in the last 1 year

APPENDIX B

Form 1: https://docs.google.com/forms/d/e/1FAIpQLSeQChJ-M9elv3n1igX1-qHHT6nDFdnARiEGMThhUsQiQ-sxaw/viewform?usp=header

Form 2: https://docs.google.com/forms/d/e/1FAIpQLScltLqfxG85bsAQFcGz4Bjtp7vLLe5FWfW8Y9tQXTfOlvY2dA/viewform?usp=header

Form 3: https://docs.google.com/forms/d/e/1FAIpQLSc3YxXnfbYArOYWgWqVu8sLPCgK9xqh7FSC3s20mUfUhWhSiA/viewform?usp=header

Form 4: https://docs.google.com/forms/d/e/1FAIpQLScTOV9wz8uylixP27klBB_yOxotH0tLE1hA99bJSXNOApi54A/viewform?usp=header

Form 5: https://docs.google.com/forms/d/e/1FAIpQLSfiU1n4cHPBCtPwOx_aLrfI3flVfoFt78EUKTOn0ZfZHbBz0Q/viewform?usp=header

Form 6: https://docs.google.com/forms/d/e/1FAIpQLSfp0tN8obObFhQvvqVk1rDHPoR8Sx6t2mzNcINCtRR4Hu1a-Q/viewform?usp=header

Form 7: https://docs.google.com/forms/d/e/1FAIpQLSdd3tpN_0-2tKruwzw_gMIanFHL8LTOOkXnfUxG512_0By-YA/viewform?usp=header

Form 8: https://docs.google.com/forms/d/e/1FAIpQLSf7WlCBhtWIOaeQ-K8F3gVAfJneAQmQxE-Pq1iEZoGUEs7i2w/viewform?usp=header

This is to certify that
Project No.: **STAT2025-4-238256**
Project Title: **From Reactive to Proactive: A Journey towards Issue Resolution with Explainable Neural Networks in the Healing Project**
Principal Researcher: **Filipe Duarte Pereira**

according to the regulations of the Ethics Committee of NOVA IMS and MagIC Research Center this project was considered to meet the requirements of the NOVA IMS Internal Review Board, being considered **APPROVED** on 4/23/2025.

It is the Principal Researcher's responsibility to ensure that all researchers and stakeholders associated with this project are aware of the conditions of approval and which documents have been approved.

The Principal Researcher is required to notify the Ethics Committee, via amendment or progress report, of
- Any significant change to the project and the reason for that change;
- Any unforeseen events or unexpected developments that merit notification;
- The inability of the Principal Researcher to continue in that role or any other change in research personnel involved in the project.

Lisbon, 4/23/2025

NOVA IMS Ethics Committee
ethicscommittee@novaims.unl.pt

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa