Fault Analysis Using Learning Models with Model Interpretation

Justin Whatley

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

August 2021

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Justin Whatley**

Entitled: **Fault Analysis Using Learning Models with Model Interpretation**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Yann-Gaël Guéhéneuc

_____ Examiner
Dr. Tristan Glatard

_____ Examiner
Dr. Yann-Gaël Guéhéneuc

_____ Supervisor
Dr. Thomas Fevens

Approved _____
Dr. Lata Narayanan, Chair of Department

August 6, 2021

_____
Dr. Mourad Debbabi,
Dean, Gina Cody School of Engineering and Computer Science

Abstract

Fault Analysis Using Learning Models with Model Interpretation

Justin Whatley

As machine learning moves from theoretical applications in academia to promising solutions to problems across industry and healthcare, effective interpretability strategies are critically important to adoption. However, model interpretability strategies can be extended to offer more than validation for the predictions a model is making. Learned models offer a proxy for the data by capturing relationships between feature inputs and target outcomes, offering a representation that can be analysed. To that end, this work describes a fault analysis system that leverages learned models to characterize faults by using SHapley Additive exPlanations (SHAP).

In particular, this fault analysis system was designed for large structured datasets such as those available in telecommunications networks. The strategy works by forming a learned representation with tree-based models using gradient-boosting. Once a problematic sample is selected for analysis, the computationally efficient implementation of the SHAP algorithm specialized for tree-based models is employed to gauge feature contributions to the performance degradation observed in the sample. Thus, this fault analysis strategy effectively provides an explanation for the degradation in a problematic sample informed through a model-based representation of the relevance of input characteristics across contexts. An evaluation of the strategy is performed, demonstrating its reliability for structured communications data using a 4G LTE dataset.

Acknowledgments

I would like to start by thanking my supervisor Dr. Thomas Fevens, for his support and flexibility. While my path was unconventional and changed a number of times, he was always there to give me guidance and sound advice. Thomas was always game to take on new project ideas or directions, thanks to his broad interests across areas of computer science and applications of machine learning. Secondly, I would like to thank my manager in the Edge Innovation group at EXFO, Sylvain Nadeau, for allowing me to mold this project into my thesis. Sylvain is creative and always ready to discuss new ideas, he was instrumental in making this project possible.

On a more personal level, I would also like to thank my lab-mate and friend Bartek, with whom I shared great conversations, collaboration for the surgical innovation project and the model interpretability path we both took for our theses. I would like to thank my family for their love and support in everything I do. And, finally, I would like to thank my partner Nathalie, who I met in my first undergraduate degree, for her love and continued support while I unexpectedly studied for another eight years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Information systems are growing in complexity, making it increasingly difficult to analyze system faults. Faults can be described as a problematic event or occurrence in the system, but this definition might vary by domain.

Traditionally, the monitoring and evaluation of large information systems was performed using expert systems with hard-coded rules defined by professionals with expertise in the field. As the complexity of the information systems grow, however, it is increasingly difficult for experts to build sufficiently complex rules that capture interactions between different components of the system. Furthermore, evaluating such complex systems when the underlying components and relationships between them evolve compounds the difficulty of building customized rules with expert help.

A key industry affected by the need for improved fault analysis is telecommunications. The recent developments and roll-out of 5G networks not only increases the data volumes in the network to monitor for service assurance, it greatly increases the complexity of telecommunications networks where faults occur by adding an additional layer of functionality. Consequently, the hard problem of fault analysis in telecommunication networks is becoming considerably harder.

Addressing this problem of growing complexity increases the need for automation and strategies like machine learning, where successes in the automation of tasks like image classification and language translation have had large impacts. In fact, deep machine learning models are now also being extended to prediction and diagnostic systems in many domains, but the size and complexity of these machine learning models themselves reduces their inherent interpretability and, thus, limit their applications in areas where interpretable models are required for integration into existing workflows. To aid in the adoption of machine learning approaches, new strategies for model interpretability and explainability are actively developed.

The intent of this thesis is to explore and evaluate the application of a fault analysis systems

for complex information systems that leverages machine learning and state of the art model interpretability strategies. The main idea of the fault analysis strategy is to train a machine learning model to represent a complex system through its data and, from this representation, gain insight into the features associated with problems (i.e., discovered faults or a problematic region of the data distribution) through the application of model interpretability strategies. In particular, this fault analysis strategy is evaluated on real-world telecommunications data from a 4G-LTE network, allowing experts to ask questions about the data such as: "What explains low throughput in a particular region of the telecommunications network?" or "What explains dropped calls on a particular cell?".

1.1 Contributions of this Thesis

This Thesis investigates the application of modern interpretability strategies to learned models of data for use as a fault analysis. The main contribution are:

1. Providing an assessment of machine learning model interpretability strategies for their possible applications for fault analysis
2. Establishing an end-to-end strategy for fault analysis leveraging machine learning model representation and interpretation using SHAP
3. Applying and evaluating the fault analysis strategy on real-world telecommunication network data

1.2 Outline

Chapter 2 of the Thesis provides a brief overview of interpretability in machine learning. It introduces the common reasons for model interpretability, describes the types of models and inherent qualities that factor into their interpretability, and introduces model interpretability strategies including SHAP.

Chapter 3 describes the fault analysis strategy proposed in this work. It provides an overview of the general strategy taken, which includes model formation and SHAP interpretation of problematic samples. It describes an approach for analysis that uses comparisons between problematic and representative samples, distances between samples features, ranking aggregate feature importance for analysis and confidence bounding by leveraging prediction error. The implementation details and model feature selection considerations for the strategy are discussed.

Chapter 4 applies and evaluates the fault analysis strategy on a 4G-LTE Radio Access Network (RAN) dataset. First, there is a description of data characteristics and data preparation steps. What follows is an extended evaluation of the representation strategy, which includes experiments showcasing model training times and performance, the application of representative control samples, representation stability, and analysis on seen vs unseen data. Visualization strategies developed for the fault analysis system, which included aggregate feature importance by sample impact and confidence bounding, are then shown and an qualitative assessment with telecommunications experts is discussed.

Chapter 5 concludes with a summary of the fault analysis approach and an evaluation of the assessment performed. Next steps and future work are discussed in closing.

Chapter 2

Model Interpretability Background

Model interpretability has become an increasingly important as complex models have moved from niche or simple utility applications, to applications with broad implications in hiring, medical diagnosis, criminal justice, self-driving vehicles, and many more. It is likely not an exaggeration to state that the interpretability of machine learning models in these domains will be regulatory requirement for most impactful application of artificial intelligence in the future.

This chapter discusses motivations for model interpretability as a strategy, the inherent interpretability of select models that can be used for fault analysis in complex systems, and introduces existing interpretability strategies and their applications to different model types.

2.1 Motivations for Model Interpretability

While interpretability and explainability are often used interchangeably, interpretability refers to the ability for us to extract *if-then* relationships in the model whereas explainability has to do with describing model outcomes in human terms. Both concepts are critical in many application of machine learning because there is often a need to unpack what the model is doing and convey the result in an understandable way.

2.1.1 The Need for Model Transparency

Consider the use of machine learning to screen for cancer in radiological images. Applying advanced machine learning strategies like deep convolutional neural networks might allow for a more reliable detection or classification of cancer than the average radiologist. However, key issues of bias, trust, consistency arise when trying to integrate such a system correctly, which will be considered below.

2.1.2 Bias

Machine learning models form a representation, or abstraction, of the data used to create them. For a supervised learning task, the model architecture and trained weights create a mapping between input parameters and a predicted outcome (i.e., target). As a consequence, the behavior of the model will lead to incorrect predictions if the data is incorrectly labelled. Returning to the radiologist example, this means that a model trained on radiological scans will have incorporated the same mistakes made by the radiologists who labelled the dataset. Of course, this can be mitigated by having multiple radiologists evaluate a scan requiring, for example, at least a 3/4 concordance between radiologist to label the image. However, once the model has incorporated mistakes from mislabelled data these can become difficult to uncover.

Any consistent mistakes in the data used to train the model, such as might occur with sampling error, would lead to model bias. What is most nefarious with bias is that consistent mistakes in sampling can affect both training and test distributions. These consistent mistakes that lead to model bias do not have to be so egregious as a mislabelling. For instance, a re-occurring artifact in the scan that was systematically present in non-cancer patients might also be present in a cancer patient scan at inference time leading the model to conclude there was no cancer because of the presence of the artifact.

Ultimately, model bias will lead to consistently incorrect predictions that can, if accepted without scrutiny, lead to consequential mistakes. Even having an unbalanced dataset, which is common across domains, leads to bias. While there will certainly be policies in place to prevent full reliance on machine learning systems when over-reliance on them may cause harm, improving predictions scores may lead to complacence and increased risk. Without being able to interpret the model itself it is difficult to form a sanity check to validate the model.

2.1.3 Consistency

The consistency of a model can be described as the reproducibility of the results produced in a new context. Another way to think about the consistency of a machine learning model in a real-world context is model generalizability, or how well the model performs for unseen data. In the real-world, however, there is usually a limit on the supply of data that can be used to validate a model. Furthermore, the test data used to validate models may have been captured in a setting too similar to the training set leading to an overestimation of the model generalizability. In the radiologist example, a model trained to detect cancer from images produced on a specific scanner may no longer perform well given images produced on a different scanner.

Using interpretability strategies allows for the exploration of the model performance for specific types of predictions. Assessing how the model predicts certain classes and what *specific input characteristics* lead to these predictions, improves confidence in the ability of the model to classify a specific class. An analysis of the model in the radiology example might be used to improve consistency by validating that specific regions of the scan correspond with the radiologist intuition of what indicates cancer.

2.1.4 Trust

Perhaps one of the greatest problems for the integration of machine learning into new areas is trust; why should the radiologist trust an opaque model inference when it is unclear how the model decides on the presence of cancer? The naive answer to this question might be that if the accuracy of a prediction from this model is, on average, better than that of a typical radiologist on a similar set of data it would be a good idea. Or that if there is a sufficiently high precision and sensitivity in the predictions it might be a good idea. However, such models often do fail when, unbeknownst to those deploying the model, newly evaluated cases fall out of distribution. In other words, models are trained in closed set environment, but then deployed in an open set environment where new examples are out of distribution.

In practice, regulatory rigor in medicine prevents the introduction of new systems like this until they are extensively vetted; trust must be earned by providing a reasonable explanation of the model inference process. To complicate this, machine learning models rely on mathematical concepts developed in computer science and statistics making some of the underlying concepts less accessible to professionals in other fields. This problem is compounded by the opacity of certain popular machine learning models, like deep neural networks. Consequently, one of the most important motivations for model interpretability strategies is unpacking the model to build trust in the decision progress being used by the model.

2.2 Inherent Interpretability of Models

The intention of the interpretability strategy, whether attempting to uncover bias, ensure consistency or build trust, will change the interpretability approach taken; there is no one-size-fits-all solution to-date. However, overarching strategies in model interpretability can still be understood and applied to address different areas of focus.

The models resulting from different machine learning strategies have properties that make them

more or less interpretable. Broadly speaking, models are split into two overarching themes: white-box and black-box models. White-box models are so titled for the simplicity of interpretation or explanation, whereas black-box models are considered to be "opaque" because of the difficulty in their interpretation. White-box models typically include the more traditional machine learning approaches, such as linear and logistic regression and simple tree-based models. Black-box models include more complex models like deep neural networks.

2.2.1 White-Box Models

White-Box Models are more directly interpretable than their counterparts, which can be explained in part by their relative architectural simplicity and the number of parameters used to represent the model. In fact, the model parameters themselves can often be associated to specific input features so they can be interpreted directly as having some level of association between the input and predicted outcome. As such, additional strategies for interpretation are not generally needed for white-box models and these are commonly favored when the goal is to have a clear explainable model.

2.2.1.1 Linear Models

Linear models such as linear regression and logistic regression are arguably the simplest cases of interpretable white-box models. When interpreting the behavior of a linear model, the weights or parameters established when training the model are a good approximation for the degree of association between the input characteristic and the outcome.

However, linear models are limited to capturing linear relationship between the features and outcomes, which can be a problem when non-linear relationships exist and, thus, more accurately represent the data. Another complication is that the parameters of a linear model are determined based on the average input set, meaning any particular feature importance is chosen based on the average contribution of the other features and not a particular case being evaluated. Thus, any influence of the context within which the features occurs is lost when interpreting linear models by their weights because there is only one representation for the *average* case.

Consequently, using linear models for their improved interpretability should include an evaluation of the reduced prediction performance of the model to assess a possible loss in representation and consider the loss of context dependence when evaluating any particular feature. Regardless, the necessity of model interpretability and explainability in applied settings and the ease of implementation will often lead to the use of these models instead of more complex ones despite a loss

in prediction performance.

2.2.1.2 Tree Based Models

Tree-based models are interpretable, white-box models that can capture non-linear relationships. Tree-based models such as decision trees and those comprising many decision trees formed through boosting (e.g., AdaBoost) and bagging (e.g., random forest) are composed of a sequence of branches that divide the dataset based on decision nodes. Different metrics can be used to split the dataset, leaving an interpretable sequence of decisions that can easily be translated to an *if-then* boolean logic.

Although tree-based approaches capture non-linear relationships better than linear models, the interpretation can be more complex, particularly in the case of large trees or aggregations of many trees. One complication arises when trying to attribute a directionality to the branch point; although the branch point gives a best indication of which inputs are important to the model, they no longer provide a simple indicator to which class a particular input is most associated. When tree-based models becomes more complex, such as in the case of tree-based models formed through gradient boosting, their interpretability may fall somewhere in between white-box and black-box models.

2.2.2 Black-Box Models

Black-box models include the many variants of artificial neural networks. The opacity of these models does not come from any restriction on observing the mode parameters but, instead, comes from the high-dimensionality of the model. For example, in a feed-forward neural network, a typical densely connected architecture will connect the nodes of each layer to every node in the subsequent layer, giving these connections a parameter weight value to describe the value of the connection. Because of this full connection between layers, the number of possible paths through the network grows exponentially with each additional layer making the interpretation of the model difficult. A trained deep-learning model will contain millions of learned parameters, making it much more difficult to associate individual parameters to features as can be done for white-box models. Consequently, although such models are effective at representing complex relationships in the world, they cannot be interpreted directly or easily.

2.3 Model Interpretability Strategies

Models that can be interpreted directly are often regarded as the best choice when model interpretability is critical for its integration into a workflow. There are key characteristics of interpretable models that form advantages and disadvantages, which will be explored here.

2.3.1 Direct Model Parameter Interpretability

2.3.1.1 Linear models

Linear models are generally the most interpretable choice for machine learning. These models establish a direct relationship between input and target values. After training a linear model, the weights (i.e., parameters) associated to feature inputs will increase proportionally to the contribution that feature input has to a particular prediction.

An important caveat is that linear models will not capture non-linear relationships and, thus, such interactions must explicitly be added to consider the interaction importance. Model mismatch is a one way to describe this problem with models of low-variance and high-bias, like linear models. High-bias models offer very *general* solutions, with predictions that will likely be somewhat correct, without being too accurate or inaccurate. While data mismatch might refer to a mismatch in the data being used to train the model not being representative of the population data, model mismatch describes a mismatch in the representation made by the model and one that would be more correct (i.e., having few spurious feature associations).

Increased reliance on features with little to no bearing on the target variable due to the model bias can result in model mismatch when linear models are fitted to non-linear data, making the interpretation incorrect. Thus, data containing more non-linear of relationships will not only lead to a decline in model accuracy for linear models, because such relationships cannot easily be represented linearly, but also reduces the overall interpretability of the linear model. This can be observed by the fraction of the model weights still attributed to features that do are not actually influencing the outcome [9].

From a computational standpoint, linear models can be difficult to use as creating a one-hot-encoding of tabular categorical columns can be infeasible in cases of high-cardinality. While this is manageable for a small dataset, as the number of feature inputs increase due to high cardinality categorical columns, interpretability will necessarily become less succinct and more difficult. A necessary strategy is then to highlight or bound the number of feature weights used to explain the model.

2.3.1.2 Simple Decision Tree

The most interpretable tree-based model is the simple decision tree, particularly when used to model an actionable set of instructions. In this model, each node represents the value (or presence, in the boolean case) of a feature with which the model can best split the training dataset. These branching points become the model and can then be applied to the test set to obtain scores to assess the accuracy of the model.

From an interpretability perspective, the value of this approach is clear. The nodes of the decision tree can be presented of a domain expert as a series of *if-then* steps, or rules to be evaluated and applied. A counter-intuitive step can easily be questioned and corrected as part of an application of such a model.

Unfortunately, the training of decision tree model is unstable, so that simple changes to the data input such as reordering or re-sampling can lead to very different sequences of branch points. To correct this, an ensemble of many decision tree models can be trained.

2.3.2 Interpretability of Complex Tree-Models

There are many algorithms used to create models based on the aggregation of trees. The random-forest algorithm is one of the most common, where many decisions trees are trained independently and the final model used is an ensemble of these decision trees. Another set of tree-based models are formed by gradient boosting algorithms. An ensemble of "weak" learners (e.g., smaller decision trees), where new learners are added based on the reduction in error they provide for the entire ensemble.

While the implementation details vary, a model formed by the aggregation of many trees reduces the instability of the model relative to the simple decision tree case, but has some negative consequences on the inherent interpretability. Aggregate tree-based models are not directly interpretable like the weights for linear models or branch points for simple decision trees, different strategies are employed to interpret feature importance. For example, the popular gradient boosting library XGBoost [2] includes three options for feature importance measurement: weight, cover, and gain. While it might be more appropriate to describe complex tree-based models as "grey-box", interpretability strategies that are necessary for black-box like LIME and SHAP can also be applied to them; these strategies will be introduced shortly.

2.3.2.1 Classic Approaches for Interpreting Trees

There are three classic approaches to evaluating tree-based models:

- Weight is a feature frequency measure across trees, telling you how many times a particular feature was used to split the dataset across the ensemble of trees composing the model.
- Cover is a measure of the feature frequency (i.e., weight) divided by the number of examples that are actually split by using these nodes as branching decisions.
- Gain, also known as Gini importance, is a measure of the average loss reduction produced by using these features as decision nodes across the ensemble.

All three of these more common feature-interpretability strategies fall prey to *inconsistency*: increased reliance of the model on a feature can *decrease* attributed importance. Further, of the three classic approaches only gain satisfies the *accuracy* property: the sum of all feature importances add up to the total importance in the model. Without guarantying accuracy it is not possible to rely on the relative contributions of features to their overall importance in the model. SHAP was designed explicitly to address these problems [10, 9].

2.3.3 Model Agnostic Strategies

To unpack some of the complexity of black-box models, local surrogate models are used to approximate the model for a subset of the data. Surrogate models work by substituting a simpler, more interpretable model to the original to explain a local set of examples. Two approaches for local model interpretability are Local Interpretable Model-Agnostic Explanations (LIME) and Shapley Additive Explanations (SHAP), which unifies LIME and game theoretic Shapley values.

2.3.4 Local Explanations

Complex models cannot be explained by summarizing their behavior without compromising or over-simplifying the explanation. However, isolating the model behavior for a specific example and approximating this behavior with a more explainable model (i.e., white-box model) allows for better explanations. Such an approach can also be extended from single examples to a subset of examples with characteristics of interest.

One strategy using this approach is Local Interpretable Model-Agnostic Explanations (LIME), using a surrogate linear model for a local explanation [15]. This is accomplished by training a local-surrogate model over a set of examples, using the prediction of the complex model as a ground truth for the simpler, explainable model. This is a useful approach when looking to really understand a subset or region of examples. SHAP expands on these ideas, incorporating Shapley values.

2.3.5 Additive Explanations

In the paper [10], Lundberg and Lee proposed a unification of different model interpretation approaches. Along with the idea of local explanations described in the previous subsection, additive explanations were used to connect different interpretability strategies for models.

From the paper, let f be the original model and g an explanation model, where the aim is to explain a local prediction $f(x)$ with input x. Simplified inputs x' are used for the explanation model with the mapping $x = h_x(x')$. The aim of a local method is to ensure $g(z') \approx f(h_x(z'))$ when $z' \approx x'$. In an additive explanation model that is composed of a linear function of binary variables we get:

$$g(z') = \phi_0 + \sum_{j=1}^{m} \phi_j z'_j \tag{1}$$

where z', m is the number of simplified input features, and $\phi_i \in \mathbb{R}$ [10]

In an additive explanation model g for the prediction, z', is a vector indicating whether the feature is absent or present such that $z' \in \{0,1\}^m$. ϕ_0 is the base rate, or expectation of the model without any additional feature information

Many model explanation strategies take the form of an additive explanation. For example, you can find explanations of this form in LIME [15], Shapley regression values [8], Shapley sampling [18], Saabas [16], to name a few. SHapley Additive exPlanations (SHAP), introduced in the work [10] is another of these approaches.

SHAP is an interpretability strategy that provides an additive explanation of a machine learning model features by approximating Shapley values [17] to fairly distribute the effect of a set of features on a model prediction [10]. Shapley values of features are computed by finding the marginal contributions when adding this feature to all possible subsets of other features to establish a fair contribution from each feature to a final payout (i.e., divergence in the prediction from the mean).

The intuitive formulation for computing Shapley values is the following:

$$\phi_j(val) = \frac{1}{\text{number of features}} \sum_{\text{coalitions excluding } j} \frac{\text{marginal contribution of } j \text{ to coalition}}{\text{number of coalitions excluding } j \text{ of this size}} \tag{2}$$

To determine the Shapley value $\phi_j(val)$, or *effect* for the model feature j: The *coalition* represents the power set of feature combinations that exclude the feature j [13]. Summing the marginal contribution of j to each coalition of other features while weighting it by the number of features in that coalition gives the Shapley value for that feature (i.e., the SHAP value).

To produce an additive explanation, an approximation of Shapley values is computed for all features m. Thus, SHAP combines the ideas of LIME and Shapley values to optimally assigns credit to different features for a particular example, offering a local explanation using Shapley values.

Given an adequate sample size, such a distribution of SHAP results can extend local explanations to characterize entire samples or subsets by calculating an aggregate result. Shapley values can offer great insight into the feature contributions to particular predictions. The size of the sample is somewhat restrictive, however, because the evaluation of many feature permutations is computationally expensive. However, the fair credit assignment for different feature importances ensures that this interpretability strategy does not succumb to the failures discussed in accuracy and inconsistency discussed in other strategies.

From a practical standpoint, there is an efficient implementation for tree-based models called TreeSHAP [9]. This algorithm is more feasible for large scale structured data analysis, offering improved computational complexity $O(TLD^2)$ relative to the general case of $O(TL2^m)$. Here T represents the number of trees, L the number of leaves, and D the largest depth. This special case formula relies on conditional expectations instead of marginal expectations, which can for produce non-zero feature attributions even when this feature does not contribute to a prediction [19, 5]. This version of SHAP is used for fault analysis in the thesis.

2.4 Assumptions About Causality

When considering the model and a possible interpretation for model predictions, it is important to differentiate the interpretation of the model itself and any conclusions about causal relationships between the input and outputs of the data. While a correlational fault analysis informed by the interpretation of a learning model can be useful in performing an analysis and exploring the circumstances surrounding a fault it will not guaranty causal relationships. This is because a machine learning model does not learn on the basis of causal relationships and, instead, learns by co-occurrence between the input and target variables during training. Thus, it is not correct to make causal claims about the model interpretation as discussed below.

2.4.1 Direction of Causal Relationship

Machine learning models, and statistical approaches more generally, establish a relationship between A and B that is not directional. That a model A predicts B may indicate that B causes

A, complicating the interpretation of machine learning models using any interpretability strategy. Correct interpretation will, thus, depend on an understood representation of the concepts which can be codified in a causal-graph. Alternatively, a temporal difference in (i.e., A precedes B) could establish directionality to this particular relationship. However, this still cannot ensure a causal relationship.

2.4.2 Third Variable Problem

By forming a testable, predictive model, there is certainly temptation to use causal arguments. However, assuming the model uncovers that A predicts B, there is no guarantee that a third property C is not causing the changes in both A and B. In fact, it is entirely possible that the input that offers a true causal relationship with the outcome may be missing entirely from the data.

Including more relevant inputs prior to training might improve the chances that the model captures true causal relationship. There is, however, no guarantee that an increased feature set will include this causal features or that the model will rely on the correct input that is the causal driver. While expanding the input sets improves the chances of including the appropriate causal features, this inclusion of many features may make the problem computationally intractable and is likely to increase reliance on less interpretable models and dimensionality reduction techniques that obscure the analysis.

2.4.3 Restricting Inputs for Causal Approximations

The limitation in uncovering causality with predictive models, and indeed most conclusions made outside of rigorous experiments and counterfactual modelling, must be put in focus if the aim is to derive important properties from a predictive model that are relevant to the problem at hand.

A validation with unseen data is necessary, but not sufficient to indicate that the features are causal properties of the outcome. To quote the British statistician George E. P. Box,"All models are wrong, but some are useful." The aim of the following research is to extract some utility from machine learning models, while falling short of establishing true causal relationships. Instead, by starting with a general understanding of relationships in the data when selecting inputs and targets to structure the model, the aim is to create an analytical strategy to guide further exploration and provide select actionable insights. Future work may incorporate causal approaches that aim to address these shortcomings.

Chapter 3

Fault Analysis System

The motivation to apply model interpretability in this project are to evaluate their implications and to develop a fault analysis system for large structured datasets. No solution to model interpretability is perfect, however, but the decisions made for model selection and interpretability strategies will be explained for fault analysis in this chapter. If the aims of the project were different (e.g., to detect bias against a particular group), different decisions might be necessary.

Although the feature contributions might be determined by any local and additive explanation method, SHAP was used for the favorable properties offered when approximating Shapley Values. Thus, in the following discussion the interpretability strategy used was SHAP.

3.1 Overview of Strategy

This fault analysis strategy can be outlined using the following steps:

1. Create a model by learning a representation of a dataset, possibly augmented by including various data sources to better represent the state of the information system, to associate inputs to outcomes of interest

2. Assess the representation of the model to unseen data to confirm that representation generalizes and, thus, can be extended to an analysis of unseen data (optionally in two steps to 1) validate generalizability, then 2) incorporate *all* data in learned representation)

3. Perform an analysis of sample deemed problematic by domain experts or predefined rules, by using model interpretability strategies (e.g., SHAP) on the trained model and using the interpretation to provide characteristic (i.e., feature or feature-value) contributions to problems in the sample. By using a model that provides local interpretations such as SHAP,

the system can be used to frame specific questions within a context; the sample is compared against a baseline, control, where divergence between the relative importances of sample features can be quantified to highlight problematic characteristics

4. Visualize impact and severity of characteristic contributions by taking the distributions or aggregates of characteristic contribution values as they relate to a problem or degradation in the problematic sample, whether it is a particular fixed outcome or the magnitude of the degradation

These steps and their relationships can be seen in Figure 1.

3.1.1 Analysis by Contrasting Samples

Analysis tools commonly allow professionals to investigate and interact with data in near real-time and it may not be practical to form specific question before starting the analysis. As such, the success of this fault analysis approaches hinges on its ability to provide a good representation of the data to allow investigation and on the responsiveness of the analytics system as an application. Satisfying both conditions can be prohibitive with modern learning models as there is a large computational cost to accurately represent large amounts of data and complex interactions.

To minimize the computational cost without losing the potential of a detailed representation, we propose forming a model on a dataset that broadly represents the information system to ensure the representation of different contexts prior to posing specific questions in an analysis of a fault. The alternative, where models are trained to new, filtered set of inputs following whenever new questions are posed by the analyst may:

- Reduce the model representation power when examples are excluded for not satisfying all constraints despite containing features valuable for the representation
- Lengthen the analysis process by forcing the retraining of models on new set of data satisfying all constraints

In our strategy we perform an analysis by contrasting samples, a general model is trained to form a representation of the inputs to the outcome of interest. Then, a problematic sample is chosen by the analyst for investigation using preconceived notions of what is considered to be problematic (e.g., low values for the target variable accompanied by other characteristics the analyst wants to constrain). Finally, the problematic sample can be evaluated for characteristic contribution with SHAP without the selection of a baseline, which is equivalent to a random baseline of the dataset,

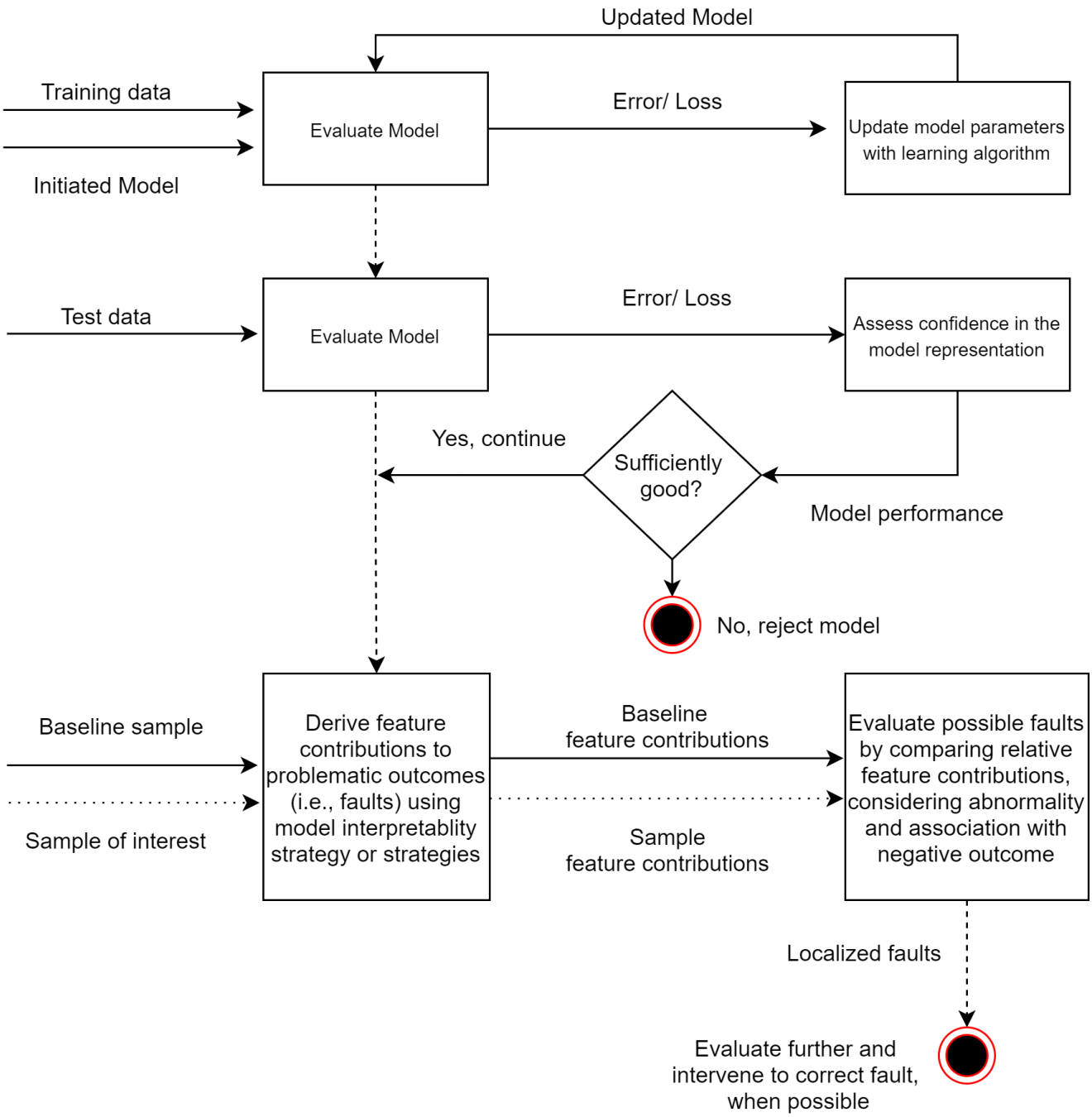

Figure 1: High-level strategy for fault analysis system that uses a model interpretability based approach

or compared against a representative baseline with appropriate constraints for that sample which is the strategy of contrasting samples. Using a representative baseline may change the distribution of expected feature attributions and offer a more appropriate comparison for the constrained features and other features that depend on them based on the new feature context.

Thus, the following strategy could be used by domain experts to ask specific questions of the data while controlling for known differences based on data characteristics:

1. Train a model to learn a representation of the dataset
2. Select a problematic samples by asking a question based on applied filters (e.g., what explains the worst performing examples for the target metric?)
3. Select a representative baseline sample based on characteristics to control for in an analysis (e.g., in telecommunication this could be the geographical region, user characteristics, cell characteristics, etc.)
4. Compare the contribution of different characteristics obtained for both samples to obtain a relative contribution to the model predicted outcome

3.1.2 Sample Comparisons

When making a comparison between samples, the goal is to quantify the divergence to determine whether the problematic sample is meaningfully different from a control sample. This quantification can be thought of as a fingerprint that represents the composition general characteristics of the sample relative to the baseline. Such a strategy might allow the grouping of specific samples to characteristics bins. These bins could later be qualified by professionals who understand the relevance of the composition of problematic features. Furthermore, this can aid in the automation of a analysis as fingerprints that include a composition of different problematic characteristics and metrics might be linked to boarder problems for the problematic sample.

A natural approach for such a comparison would be the global evaluation of SHAP values between a problematic sample and control with Kullback-Leibler (KL) Divergence. This approach could be used to gauge the absolute divergence between the problematic and control samples, irrespective of which particular feature diverged. However, because of the diversity of features and implications of particular features in the analysis such a generic divergence may not be all that useful in deciding whether a particular sample is worth exploring; KL divergence will only indicate how different the samples are, not the way they are different which is needed for analysis. Instead,

the approach used for sample comparison will still focus on the distribution of SHAP values at a feature level.

To perform sample comparisons, the SHAP distributions of each feature is compared by computing the first Wasserstein distance (also known as Earth-Movers distance) to represent a movement of probability mass between the two samples:

$$w_1(\mu, \nu) = \inf_{\pi \in \Gamma(\mu,\nu)} \int_{R \times R} |x - y| d\pi(x, y) \tag{3}$$

Where $\Gamma(\mu, \nu)$ represents the set of probability distributions for the computed SHAP values in $R \times R$ with marginals μ and ν.

The approach was taken because certain distributions will not be regular and, if they are, may be highly skewed if the target variable used is skewed as is shown in later experiments. For normally distributed samples of SHAP values, standard statistical comparisons of normal distributions might be appropriate. However, because of the transformations by the model to create SHAP values, our preference was for this non-parametric approach.

To assess the direction of this distance, the two distributions are compared in aggregate. Comparing the medians of the two SHAP distributions of features gives the direction of their relative importances in association to the target variable. For examples, given a median feature SHAP value of *-100* in a problematic sample and *0* in a control sample, the sign indicates an aggregate negative contribution for the feature relative to the control. While this median comparison might be used directly, the w_1 provides a better representation of the actual distance between the SHAP distribution masses. In particular, w_1 of the SHAP values provide a better quantification of the distances between the contributing features in the sample of interest relative to an appropriate control.

3.1.3 Importance Ranking

When evaluating characteristics (i.e., features, or feature-values) associated with outcome predictions using SHAP, an internal internal ranking is helpful to bring the most relevant characteristics to the fore. Getting the Wasserstein distance w_1 of feature importances distributions *between* samples, as in the sample comparison section, is useful in determining how different the values are from what would be expected with a representative control. An internal evaluation *within* the sample that ranks the relative features contributions to the negative outcome is determined using aggregate SHAP results.

For a fault-analysis system, this ranking would be made on the contribution in the direction of the problematic metric (e.g., those characteristics most associated to lower performance). The

relative contributions in the direction of the problem can be summarized and ranked for features by Algorithm 1 as seen later in results Figure 11 and feature-values by Algorithm 2 as seen later in results Figure 10 and Figure 12.

Algorithm 1 Feature contribution rankings

Input: Data matrix $S \in \mathbb{R}^{n \times m}$ of SHAP results in a sample of size n with m features.

Output: Average feature-level contribution vector f

1: **for** $j := 1$ **to** m **do**:
2: $\quad f_j \leftarrow \sum S_{*,j}/n$ ▷ Get average SHAP contribution of feature-value bin
3: **end for**
4: **return** $argsort(f)$ ▷ Return aggregate contributions of features to outcome

Both these algorithms provide a sorted average importance for the characteristics in the sample at a different level of granularity. The high-level Algorithm 1 shows a feature level view that shows the average SHAP values of any particular column of the dataset. The lower-level Algorithm 2 produces bins for k quantiles and each category instance, to provide improved visibility on the importances of specific characteristics. Together, these rankings provide an initial order of priority, or points of focus for the analyst reviewing the results.

SHAP estimates the contribution of a feature (i.e., Shapley value) in $E_X\left(\hat{f}(X)\right)$, but the expected value may be different given a new, representative baseline. Thus, adjustments to the feature importances based on the representative baseline *mean* will be necessary to leverage this difference. An external comparison between aggregate SHAP values could be made by comparing the distribution of SHAP values for a particular characteristic in the problematic sample to the control sample. This distribution comparison will still allow ranking and quantification with w_1 of the differences between SHAP value distributions for any characteristic.

3.1.4 Considering Model Prediction Accuracy

The Shapley values estimated with SHAP provide a relative importance for each feature giving their contribution to a prediction made by the model. The goal of the fault analysis system described in this Thesis is to evaluate samples subject to a fault, indicating the characteristics most associated with problems in the sample. This sample is commonly framed or thought of as a question such as: "Given a particular region, what explains the poor throughput observed in a telecommunication network?". The answer can be better characterized through the use of a model with a representation of a wider area of the telecommunication network rather than an observation

Algorithm 2 Feature-value binned contribution rankings

Input: Data matrix $S \in \mathbb{R}^{n \times m}$ of SHAP values (n points with m features), data set $\mathbb{D}$ and original set of features D, number of bins q (for quantile-based discretization).

Output: Average feature-value level SHAP contribution f_{bin}

for d **in** D **do**:
  if $is\_numerical(d)$ **then**: ▷ Numerical case
    $interval_1 \dots interval_q \leftarrow quantile\_based\_discretization(\mathbb{D}_d, q)$
    for $l \coloneqq 1$ **to** q **do**:
      $s\_bin_l \leftarrow match\_rows(interval_l, S_d)$ ▷ Find corresponding rows in SHAP values
      $f[d]\_bin_l \leftarrow \sum_{S_{ij} \in s\_bin_l} S_{ij} / |s\_bin_l|$ ▷ Average SHAP by quantile interval bin
    end for
  else: ▷ Categorical case
    $value_1 \dots value_k \leftarrow get\_unique\_values(\mathbb{D}_d)$
    for $l \coloneqq 1$ **to** k **do**:
      $s_l \leftarrow match\_rows(value_l, S_d)$ ▷ Find corresponding rows in SHAP values
      $f[d]\_bin_l \leftarrow \sum_{S_{ij} \in s_l} S_{ij} / |s_l|$ ▷ Average SHAP by category instance bin
    end for
  end if
end for
return $argsort(f\_bin)$ ▷ Return average SHAP contributions of characteristic bins

of the local region which will contain limited information.

A problem arises in that most models, particularly models of sufficiently complex real-world data, are not perfect. As such, in making predictions there will be a difference (i.e., error) between the model predicted value and actual value of a target for any given example. The average error on a set of examples might be taken to represent the quality of the model in representing the relationships between the inputs and target variable for these data.

The described fault analysis strategy uses SHAP to *understand data* in a sample believed to have an underlying fault, that is unlike some other applications of SHAP which aim primarily to *understand the model*. Whenever a SHAP explainer function is used to evaluate an individual case, a SHAP value is given to represent the relative importance of each feature-value in the prediction made by the model. Therein lies the problem; if model error is high, the contributions of particular features to the overall degradation in a problematic sample may be misrepresented.

Thus, the importance of features as computed in Algorithm 1 and feature-values as computed in Algorithm 2 will not reveal poor predictive performance by the model on specific examples. As such, it could be helpful to account for systematic over-prediction or under-prediction in the presence of specific characteristics, as systematic error for cases in the sample may lead to the over or under representation of the importance of a specific feature. We propose providing visibility on model prediction error through confidence bounds for the contributions of individual features.

3.1.4.1 Feature Importance Confidence Bound

To address this problem we propose establishing a feature importance confidence bound heuristic with aim of using model prediction error as feedback to indicate confidence in the feature contribution to degradation of a sample. Because there is no restriction on using the ground-truth as part of an analysis system, the error between the ground-truth and model prediction are used to create confidence bounds. The confidence bounds represent deviation (i.e., error) between the model expectation when certain characteristics appear in a local sample to the actual observed target value. How to use the error as a feedback mechanism is explored below.

Reviewing the original SHAP formulation discussed in the previous chapter:

$$g(z') = \phi_0 + \sum_{j=1}^{m} \phi_j z'_j \tag{4}$$

where g is the explanation model for the prediction, (z'), where $z' \in \{0,1\}^m$ is a vector indicating whether the feature is absent or present, representing the proportion of contribution by each feature to the largest possible coalition size, which is the combination of all m features. The sum

of marginal contributions is determined for each feature to establish showing a deviation from the ϕ_0 base prediction (i.e., the expected value if no feature details were known) [13]. The explainer calculates SHAP values $[\phi_1, \phi_2, \ldots, \phi_m]$ for one example with the property:

$$\hat{y} - \phi_0 = \phi_1 + \phi_2 + \ldots + \phi_m \tag{5}$$

$$\textit{SHAP-values} = [\phi_1, \phi_2, \ldots, \phi_m] \tag{6}$$

where ϕ_j is the effect of each feature on the prediction of the model, $\hat{y}$ is the model prediction and ϕ_0 is the expected value for the model.

To distribute the error between the different features evenly to create confidence bounds, the following formula could be applied:

$$y + \epsilon = \hat{y} \tag{7}$$

$$y - \phi_0 + \epsilon = \phi_1 + \phi_2 + \ldots + \phi_m \tag{8}$$

$$y - \phi_0 = \phi_1 - \epsilon + \phi_2 - \epsilon + \ldots + \phi_m - \epsilon \tag{9}$$

$$y - \phi_0 = \phi_1'' + \phi_2'' + \ldots + \phi_m'' \tag{10}$$

$$\textit{confidence-bounds} = [\phi_1'', \phi_2'', \ldots, \phi_m''] \tag{11}$$

This distribution of error assumes that *each* feature can at most be responsible for the entire error ϵ, as a worst-case error for selection ϕ_j. However, an assumption that each feature contributed equally to the error seems unreasonable given: 1) a model that has learned reasonable representations of relationships between the inputs and a predicted value and 2) SHAP indicates *how much* each feature is relied on by the model for prediction and, therefore, a feature that is not used should not be responsible for any of the error. A similar argument can be made when, instead of making a worst-case assumption, dividing the error evenly between each features.

Using the assumption that if credit is fairly distributed, so is blame; we propose the following heuristic. Making the assumption that SHAP credit and blame should be applied proportionally

based on feature attribution, we incorporate local error proportionally to the feature contribution by substituting the $prediction_\delta$ with the $ground\_truth_\delta$ in the following way:

$$prediction_\delta = \delta(\hat{y}, \phi_0) \tag{12}$$

$$ground\_truth_\delta = \delta(y, \phi_0) \tag{13}$$

$$\hat{y} - \phi_0 = \phi_1 + \phi_2 + \ldots + \phi_m \tag{14}$$

$$(\hat{y} - \phi_0)/prediction_\delta = \phi_1/prediction_\delta + \phi_2/prediction_\delta + \ldots + \phi_m/prediction_\delta \tag{15}$$

$$1 = \phi_1' + \phi_2' + \ldots + \phi_m' \tag{16}$$

where ϕ_j' is the fraction of contribution each feature is responsible for in the prediction.

Applying these same fractions to the ground-truth values we obtain the following equation to scale the original SHAP values:

$$ground\_truth_\delta = \phi_1' * ground\_truth_\delta + \phi_2' * ground\_truth_\delta + \ldots + \phi_m' * ground\_truth_\delta \tag{17}$$

$$ground\_truth_\delta = \phi_1'' + \phi_2'' + \ldots + \phi_m'' \tag{18}$$

$$confidence\text{-}bounds = [\phi_1'', \phi_2{}'', \ldots, \phi_m''] \tag{19}$$

where ϕ_j'' represents the confidence bound calculated by scaling the original SHAP values ϕ_j, that sum to the $prediction_\delta$ to confidence bound values ϕ_1'' which sum to $ground\_truth_\delta$. These confidence bounds can be extended to an entire sample using Algorithms 1 and 2.

The feedback is not sufficient to justify replacing the model importances established by SHAP, as the model was likely formed using far more data than is available in the sample. However, it serves to provide visibility on model error that might occur systematically in the presence of specific characteristics. Although we ultimately cannot know whether the error represents noise in the measurements or contributions from unknown variables, providing visibility on systematic prediction error when certain feature are present may reinforce or instill doubt for the interpreted importance provided by SHAP when performing analysis on a problematic sample.

There is a serious drawback to implementing this strategy with SHAP values to a general case: when the predicted value and ground truth value are on opposite sides of ϕ_0, calculating confidence bounds this will effectively inverse the sign and, therefore the relationship provided by SHAP. This does not make any intuitive sense given a small error near the mean. However, for the special case of fault analysis when values are consistently and often considerably below the ϕ_0, this strategy can still be used to provide confidence bounds for the most problematic variables. However, any heuristic implementing this strategy must prevent these sign changes to produce a reasonable result.

3.2 Implementation

3.2.1 Model Selection

As discussed previously, linear models stand out for their off-the-shelf interpretability making these a simple way to identify more obvious problematic metrics in the network. However, the size of the dataset and the presence of high cardinality categorical features can make the application of linear models prohibitive. For instance, one-hot-encoding datasets with high cardinality to allow the creation of a linear model is possible, but the memory and computational constraints of transforming the data to obtain thousands of new feature columns make the use of linear models highly impractical for high cardinality data.

Neural-networks offer state of the art prediction performance at the expense of interpretability because of the high dimensionality of the models. Neural-networks were not used for this project because the goal was to create a fault analysis system where interpretability is paramount. Further, for structured data, neural-network approach may not offer much of a prediction benefit relative to simpler models like tree-based models formed using gradient boosting.

Tree-based models formed using gradient boosting are well suited for computing large amounts of structured data [2] and, thus, will be used for the fault analysis strategy. Using prediction performance as a proxy for the quality of the representation, tree-based models are a good choice to improve prediction performance relative to simpler linear models given sufficiently complex data and a comparable choice to neural networks. In particular, a gradient boosting library called LightGBM is highly optimized for the training of tree-based models using gradient boosting [6], allowing parallel computation and direct support for categorical variables [3].

3.2.2 Interpretability Strategy Selection

While their are computational considerations, SHAP distinguished itself as a clear choice among the available interpretability strategies discussed in Chapter 2. It offers accuracy and consistency guaranties when evaluating feature characteristics that other model interpretability strategies like Gain, Permutation, and Saabas do not [10].

SHAP can be used in a model agnostic way, which means that implementation exists for different models. However, there is a computational constraint on applying this strategy to evaluate complex models because of the combinatorial nature of Shapley value. This computational constraint is particularly important in a fault analysis system when the goal is to provide a analysis relatively quickly in a space where conventional tools operate in near real time. The general approach used to compute SHAP must use approximation because the summation over all possible feature-subsets, and generating all possible permutations in general, is NP-hard [12]. However, in the tree-based SHAP computation introduced earlier, Lundberg *et al.* devised a way to compute exact local explanations using Shapley values on tree-based models in polynomial time [9]. The improved time-complexity when computing SHAP to interpret feature attributions in a model further justifies the use of tree-based models over neural network models. Thus, the TreeExplainer version of SHAP will be used for the fault analysis system to evaluate tree-based models.

3.2.3 Feature Selection

The selection of features is a particularly difficult in forming a model for fault analysis, with no one-size-fits-all solution. Because machine learning models are produced by co-occurrence as opposed to causal relationships, the features selected to be part of the model are critical in determining the result produced by the analysis.

3.2.3.1 Saturation

One key consideration is that including features too highly associated with the target variable prevents the discovery of association of *other* features with the target. This is because the model will use the highly predictive feature at the exclusion of the others, saturating the signal for a single feature. For instance, given a feature that is sufficient to predict 99% of target outcomes on its own, the model will not put much weight on other input features because they are not necessary for the prediction. However, assuming no simple answer in a complex information system, an analysis that provides a variety of important features for further exploration will be more useful to the domain expert.

Saturation is a problem for the fault analysis system because allowing multiple input features may provide a more nuanced description of the fault. Ideally, different inputs will be used in different contexts, depending on the goal of the analysis. Discovering such a highly associated feature that is used at the exclusion of other features may indicate a data leakage (i.e., the feature is simply another descriptor of the target). Removing these features may be required to produce a nuanced or informative story from the remaining features.

3.2.3.2 Metrics vs. Dimensions

Another challenge in establishing the fault analysis system is deciding the type of data to include in the model. A dimension, or instance of a category, might offer a characteristic location for the fault. While a dimension can offer a location to point to, indicating *where* the fault is occurring, the actual problem can be more nuanced and only appear given conditions described by metrics. Metrics can describe the network state for a sample, but do not offer much actionable insight when the problem is not localized.

In a telecommunications example, perhaps there is a problem with particular equipment only when there is a certain load on the network. Ignoring the load on the network might not allow the model to properly account for the requirement that *both* characteristics appear before predicting the outcome for an end-user of the network. In a second scenario, imagine a metric is highly predictive of an outcome for an end-user of the network. Without an associated dimension to localize the problem, this metric information may not provide enough useful or actionable information without a proper analysis and further investigation. Consequently, a careful selection of the included features is required to create an effective interpretation of the model for fault analysis. This is explored in the following sections.

3.2.3.3 General Case for Feature Selection Choice

Understanding what the machine learning model represents is critical to effectively use the model for fault analysis. Fundamentally, a machine learning model is a function that maps inputs to a target outcome, learning associations based on the frequency of co-occurrence in the training dataset. A framework for the implications for the fault analysis strategy can be seen in Figure 2.

The left hand side of the figure breaks down the input choices that comprise feature selection, which can be metrics (i.e., key performance indicators), descriptive characteristics, and actionable characteristics. The model in the middle of the figure represents the mapping. The right hand side represents possible outcomes which can be a categorical outcome (i.e., classification) or a

continuous value (i.e., regression).

Careful selection of input features for a particular outcome is important depending on the ultimate goal of the fault analysis system. The implications of these choices are explored in the next section. Note that correlations between input features will also have large implications in what the model relies on for prediction and is relevant to feature selection but this is not explicitly discussed here.

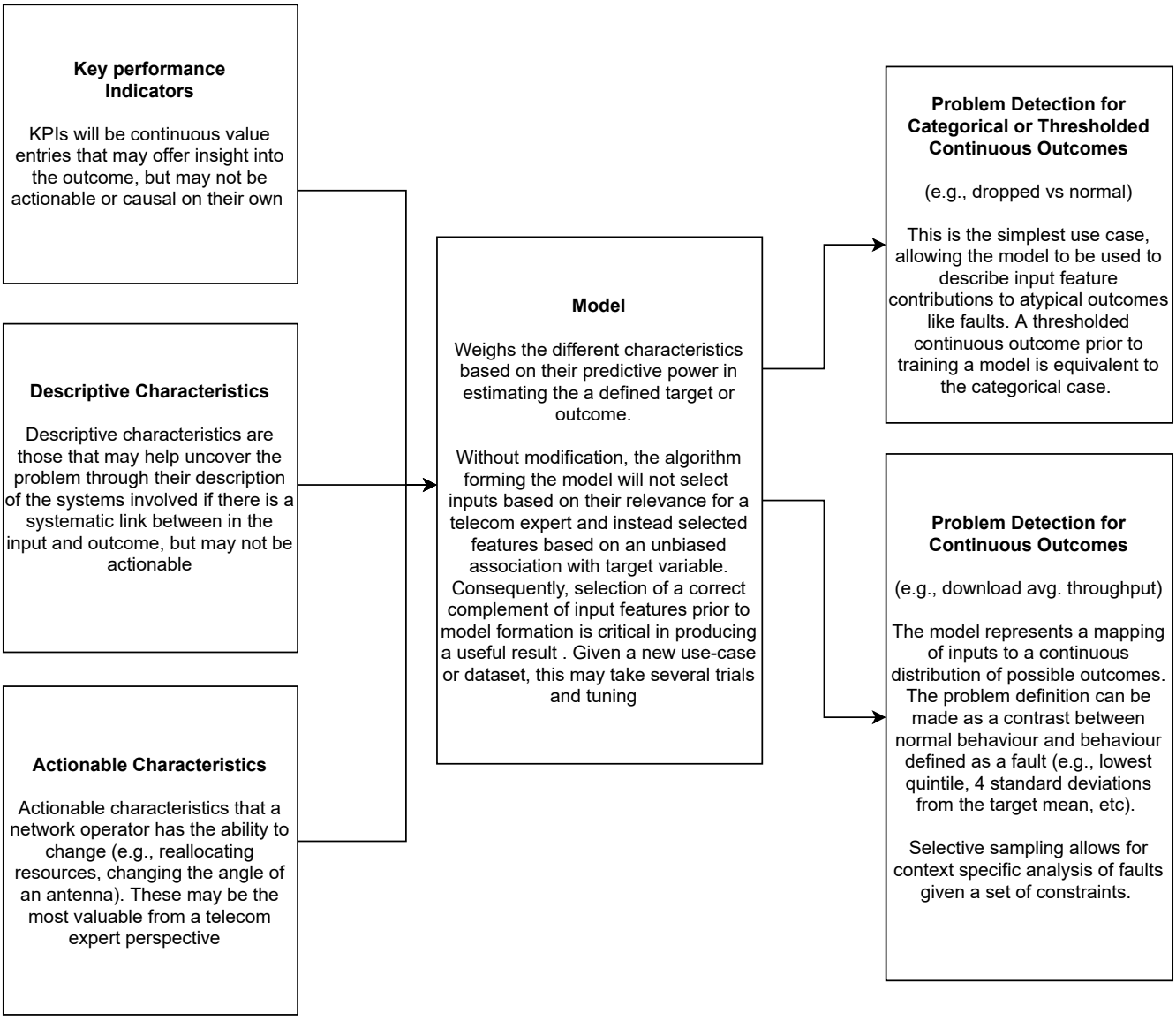

Figure 2: General description of data characteristics that may be present in a measurement system

3.2.3.4 Feature Selection by Use Case

This section discusses some of the high-level implications of having different feature composition from the left hand side of Figure 2.

The best fault analysis use case might be crafted through a combination of different KPIs, descriptive and actionable characteristics. Furthermore, it is entirely possible that certain fields will not fit neatly into these input groups already defined. The purpose of this exploration is to consider the complexity and implications of using certain input features over others and how this might impact the fault analysis system based on what the model is learning about the data.

Figure 3 outlines a system using all described input characteristic types. In this formulation, key considerations might be:

- Adding too many features may reduce interpretability of the result by providing an interpretation too complicated for the individual evaluating the result
- Highlighted symptoms may obfuscate possible causal contributors to the fault, as the model will rely on feature inputs based solely on correlations with the target variable

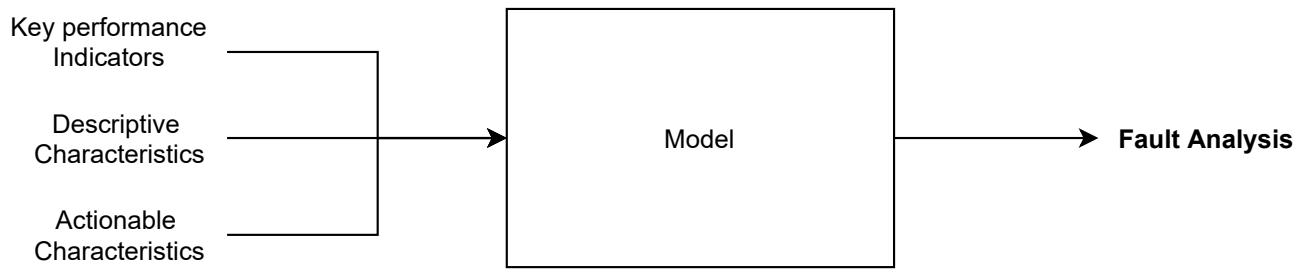

Figure 3: A fault analysis system that uses all input types introduced for analysis

Figure 4 outlines a system that avoids performance metrics in the model, instead relying only on the outcome as the symptom. Thus, the descriptive and actionable characteristics can be localized and directly associated to the outcome during fault analysis. In this formulation, key properties might be:

- Provides a focused picture of the characteristics associated with the problem that aid in localizing the issue
- There may not be enough information in the model representation to correctly predict outcomes and, thus, provide an insufficient analysis

Analysis without performance measurements or KPIs

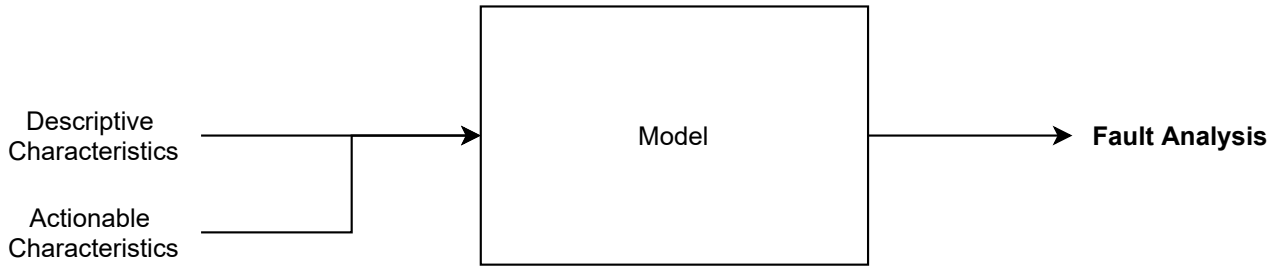

Figure 4: A fault analysis system that avoids KPI metrics as potential symptoms

Figure 5 outlines a system that only includes actionable inputs, so that any feature input with sufficient impact on the outcome will allow the analyst to take action. In this formulation, key properties might be:

- Provides the most actionable analysis (e.g., experts can intervene if any feature is deemed important by the model)
- May not accurately predict the outcome due to insufficient context
- Although actions may appear clear, incomplete information due to missing context features may lead to overconfidence in a particular action which is problematic as causal relationships are not established by this fault analysis strategy

Analysis for actionable insight only

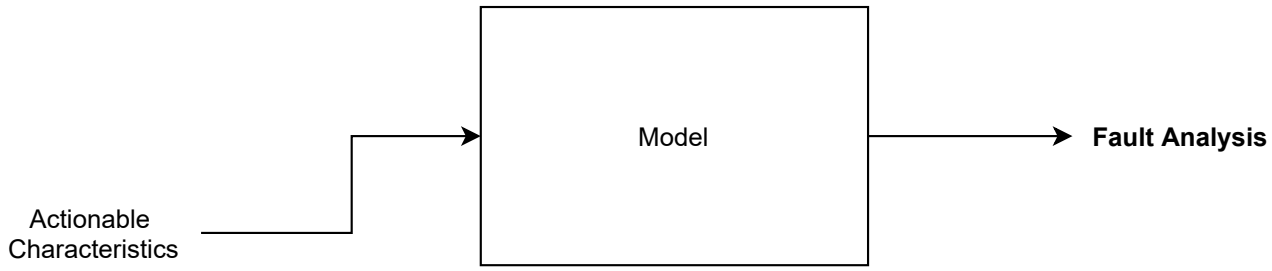

Figure 5: A fault analysis system that only includes actionable inputs

Figure 6 outlines a system that only includes KPIs and actionable characteristics, to provide a performance context within to qualify when actions are necessary. In this formulation, key properties might be:

- Added KPIs may help qualify the actionable characteristics through interaction effects

Analysis for qualified actionable insight

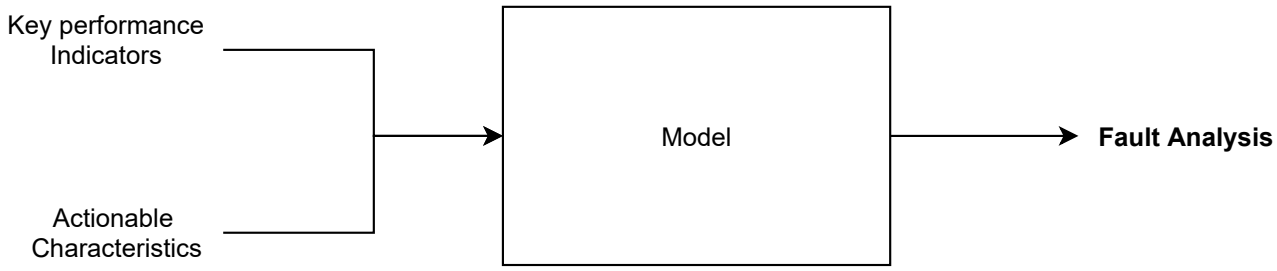

Figure 6: A fault analysis system that ignores non-actionable descriptive characteristics

Chapter 4

Assessment

The analysis strategy discussed in this Thesis was developed through several iterations of feedback from Radio Access Network (RAN) telecommunications experts at EXFO to converge on a presentation of the data that is sufficiently intuitive to allow experts to quickly determine analyze problematic samples of structured data.

4.1 Dataset

The primary dataset consists of telecommunication Call and data Detail Records (CDRs) for LTE radio access networks (RAN), obtained at EXFO. A CDR is a trace record produced by a passive probe in a telecommunication operators network containing metadata of an individual call or data transaction that took place over the network. Such CDRs are monitored and evaluated to ensure a high quality of experience for end-users of the network, by aiding distributions of cell resources and proper configuration of the telecommunication network [1]. For privacy, no operator or call-level details are provided and sensitive data was removed prior to modelling.

The CDRs used were generated across peak and non-peak hours of activity on the network, from 12:00 to 22:00, to model the impact of cell-level congestion on the quality of service for end-users. The goal was to assess the viability of the proposed fault analysis strategy as applied to this real-world information system in collaboration with EXFO telecommunications experts to qualify the viability of such a solution and to aid in the partial automation of fault analysis. Specifically, the use case discussed for this work maps characteristic inputs of CDRs to the measured download average throughput per CDR, which is the measured download throughput for the downlink between the antenna and the user equipment (UE).

4.1.1 Data Preparation

4.1.1.1 Enrichment

In order to evaluate across modalities, CDRs were enriched by including cell specific information. This enrichment adds cell level characteristics to the user-level characteristics found in CDRs to improve localization for particular faults.

Specifically, LTE Physical Resource Block (PRB) Utilization represents the level of utilization of a cell and can be used to measure of cell congestion. Including this information with a particular data record can help disentangle throughput performance from a user-level handset issue to broader congestion on a particular cell. Other cell characteristics used to enrich the CDRs include the number of Physical Cell Identity (PCI) clashes along with their severity, whether the cell is an overshooter, and the number of other cells interfering with this particular cell. These cell augmentations were commonly excluded from the experimental analysis for brevity, as the models trained rarely relied on them for prediction.

4.1.1.2 Data Types

Continuous metrics were left unaltered, whereas string data representing a discrete characteristic or dimension of the CDR (e.g., handset type, manufacturer) were treated as categorical instances in training [**?**].

4.1.1.3 Cell Selection

To simplify the representation without losing much important detail, only characteristics of the Cell at the end of the record (i.e., End Cell) were included. The Start Cell and End Cell remain the same for most CDRs, indicating that most records take place on a single cell. To exclude CDRs where the Start Cell and End Cell are different would offer more precise Data Enrichment, but would exclude CDRs with handovers operations from one cell to another, which are a common source of problems in causing Dropped calls in cellular networks. However, that use case will not be discussed here. The Cell discussed in future sections will always be in reference to the End Cell at the end of the CDR trace.

4.1.1.4 Data Filtering

Filtering was done to ensure that only CDRs of *data* transactions were included and that data volumes are at least *1000 bytes* to ensure that throughput was not bounded by the low data volume

of the transaction. This was done to align with current practices used by the telecommunication professionals when investigating throughput issues. The number of remaining CDRs after the application of these filters was *15,600,403*.

4.1.2 Characteristics

The download average throughput characteristic is present for all data transactions in the CDRs and distributed as seen in Figure 7.

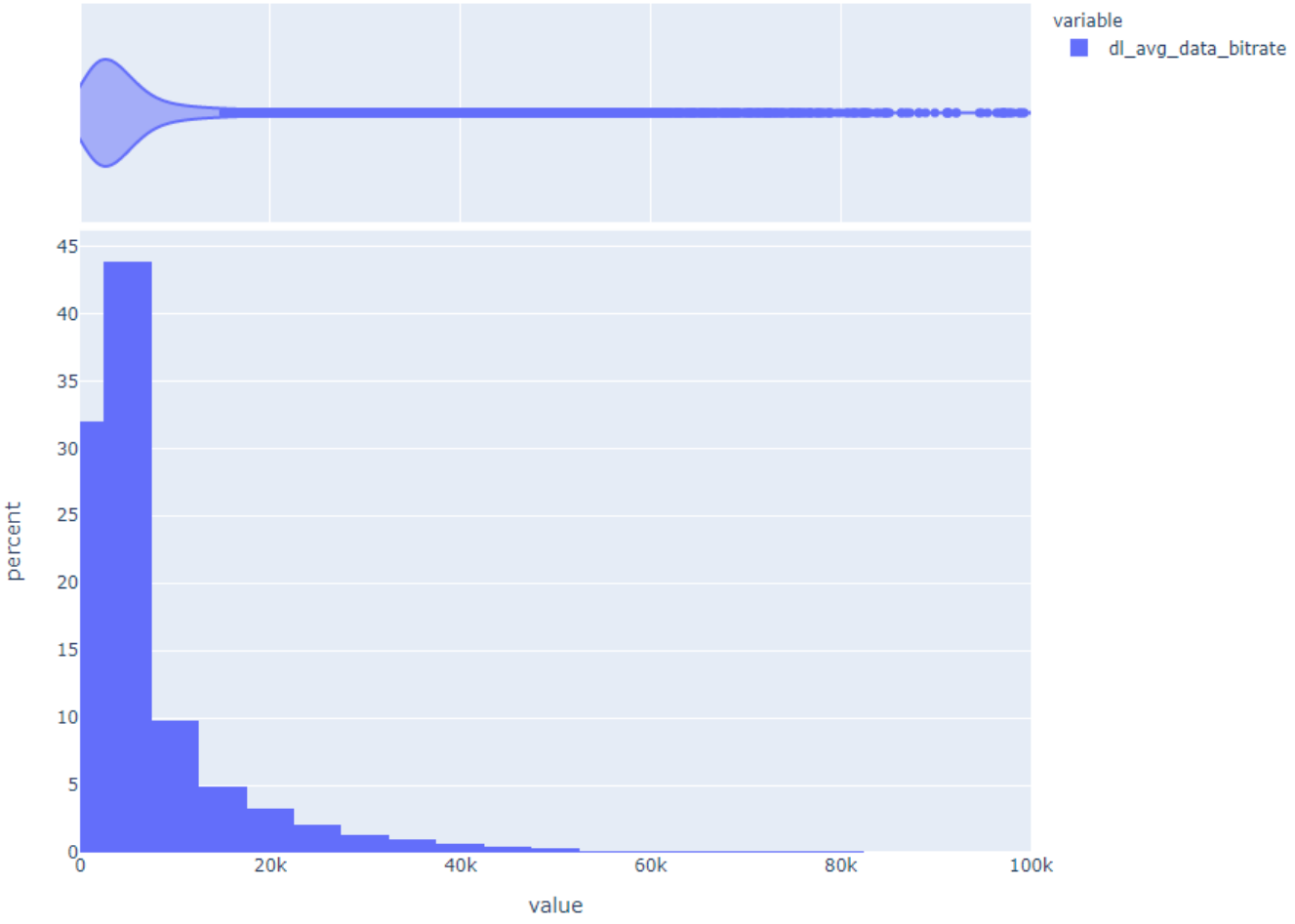

Figure 7: Average Download Throughput (kbps) violin (top) and bar (bottom) density plots of dataset

The distribution of Download Average Throughput (kbps)is highly skewed, showing large differences in mean and median results. These characteristics can be seen in Table 1. This type of

distribution can be modelled somewhat faithfully using tree-based models, but it does make it more difficult to interpret mean results. Linear correlations to continuous metrics in the dataset were measured and listed in Table 2, showing small positive correlations with Reference Signal Received Power (RSRP), Signal-to-noise Ratio (SINR), Reference Signal Received Power (RSRP), and small negative correlations with Cell physical resource block (PRB) Usage Ratio, Timing Advance to Cell.

Table 1: Basic distribution statistics for Average Download Throughput

Statistic	Score (kbps)
Median	3857.142
Mean	6579.526
Standard Deviation	8691.292
Kurtosis	19.091
Skew	3.535

Table 2: Correlations between continuous metrics and Average Download Throughput

	Correlation (r) with Avg. Download Throughput
Reference Signal Received Quality (RSRQ)	0.102255
Signal-to-noise Ratio (SINR)	0.131026
Reference Signal Received Power (RSRP)	0.141005
Cell physical resource block (PRB) Usage Ratio	-0.105429
Timing Advance to Cell	-0.067372
Download total bytes	0.162478

In the interest of space, correlations for the many categorical instances in the dataset are not shown here. However, the number of unique categories for categorical features is shown in Table 3. When the number of unique categories was sufficiently small, the possible category instances are displayed.

Table 3: Dimensionality of categorical input features

Categorical field	Count	Unique categories
EnodeB	1730	-
Cell	13984	-
Quality of Service Class Identifier (QCI)	15	-
Carrier Frequency EARFCN	4	[124, 132, 3510, 6350]
Service Used	4	[lte, originating-lte, terminating-lte, CS-fallback-provision]
Overshooter Level	4	[High, Low, Medium, None]
Manufacturer	941	-
Handset Type	4022	-
Roamer	2	[Yes, No]

4.2 Evaluation of Representation Strategy

4.2.1 Model Training

Model training times and performance were evaluated using three available training strategies in the LightGBM library: Gradient Boosting Decision Trees (GBDT), Dropouts meet Multiple Additive Regression Trees (DART) [14], and Gradient Boosting Decision Trees (GOSS)[6]. On a brief discussion of the merits for each, GBDT is the standard approach that samples new datapoints for training on the basis that they provide the highest gradient (i.e., error) for the decision trees to date. DART is purported to reduce the overspecialization found in a regular GBDT approach by introducing a form of dropout [14]. Finally, GOSS modifies the sampling strategy of GBDT by keeping all large-error samples and taking a random sampling from the remaining training instances with small error/low gradient.

While each gradient boosting strategy might have implications for the resulting model representation and feature importance attributions from SHAP, our selection was based on computation speed and model error for the dataset. Table 4 shows one such evaluation made on runtime, training accuracy, and validation accuracy for these each boosting strategy over 1,000 training iterations and otherwise constant sampling hyperparameters across 20 CPU cores. This evaluation shows that the default setting of GBDT was most appropriate for the task. When comparing GBDT with GOSS, there was no difference in the prediction error but the training time was longer. This may be the result of poor hyperparameter selection for GOSS, but this was not explored further as it was considered tangential to the project.

Table 4: Training times and prediction error based on boosting strategy with LightGBM for dataset

	Training Time (s)	Mean Abs. Error	
Boosting strategy		Training	Validation
GBDT	3516.36	3715.65	4274.62
DART	45182.25	3911.07	4279.76
GOSS	4361.08	3715.65	4274.62
GBDT (358^{th} iteration)	1258.86	3911.18	4293.59
DART (1000^{th} iteration)	45182.25	3911.07	4279.76

The introduction of dropout using DART as shown in Table 4 indicate some protection against

overfitting insofar as there is a smaller difference between training and validation errors relative to GBDT. To show this we use observations of GBDT and DART comparing the 358^{th} and $1,000^{th}$ boosting iterations, respectively, DART does appear to be somewhat protective against model overfitting on this basis, with a lower validation loss relative to training loss. However, the more than 35 fold computation cost of training a model with DART make it impractical for our fault analysis use case.

Table 5: GBDT training rounds with increasing leaves count over 1,000 boosting rounds for the entire dataset

Leaves	Time (s)		Mean Abs. Error	
	Training	SHAP	Training	Validation
250	1391.89	1492.24	4040.58	4327.37
500	1982.16	3600.00	3897.42	4303.37
750	2882.34	5948.42	3779.69	4283.00
1000	3465.94	8450.72	3715.65	4274.62
1250	3930.92	10575.79	3650.89	4267.00

In gradient boosting, new trees are added with each boosting round. The size of the final model will depend on the size of each tree added and the total number of trees composing the model. While the tree depth of individual branches is not controlled in this evaluation, Table 5 demonstrates the computational costs of increasing the size of and, based on the reduced prediction error, improving the representation of the model. SHAP evaluation times were calculated on *257,244* CDRs across 10 samples samples. These results demonstrate that SHAP computation scales linearly with the number of leaves per tree (i.e., boosting rounds).

In general, it was difficult to overfit this data as demonstrated by the fact that training and validation error continued to improve for all models (i.e., validation error did not diverge). The difficulty in overfitting this data may have to do with the dataset size, cardinality, missing variables and general noise in the system. Furthermore, the large influence in error for extreme cases with high throughput may increase the learning difficulty. However, the interpretation for a sufficient large problematic sample, ranking the learned impact of features from Shapley values should still provide insight in the factors that consistently improved model prediction.

From a practical standpoint for analysis for fault analysis, the difficulty in overfitting the data may justify the use of observed data in analysis. This difficult in overfitting indicates that the model cannot memorize the training examples and, thus, that evaluating seen data with SHAP

to provide a model based summary of importances has less of a risk of being formed spuriously. Moreover, for this specific use case, computational constraints appear to be the bottleneck for a fault analysis tool where continuing to improve the quality of the representation may reduce the practical value of the approach.

4.2.2 Using Representative Baselines

Using a representative baseline is a central component of the contrasting samples idea described in Chapter 3. For instance, when performing an analysis of a cell that appears to be problematic in a telecommunications network selecting an appropriate *representative* baseline could involve sampling CDRs from nearby cells, or other cells connected to a particular EnodeB, or cells of the same carrier frequency, or the surrounding geographical area more generally.

For example, certain cell carrier frequency bands will have physical properties that will allow better throughput transmission. Thus, selecting CDRs that were connected to a nearby cell of the same frequency may be a better choice in isolating true performance differences and, therefore, characteristics that are actually problematic. A representative control can be used to properly evaluate the importance of features that may be relevant to general performance degradation, but not necessarily relevant when compared to an appropriate baseline.

To evaluate the differences in the randomized baseline against a more representative baseline where frequency was controlled, consider Table 6. In this experiment a representative baseline is formed by randomly sampling CDRs that share the Carrier EARFCN 3510. This carrier has a bandwidth of 5Mhz, the lowest on this network.

As expected when comparing a problematic sample to a randomized baseline, there is a considerable Wassertein distance w_1 between the distribution of Shapley values when compared to a control. The metric was computed using the available SciPy package [20]. This table demonstrates that, in selecting a control baseline, there is a considerable reduction in w_1 when controlling for Carrier Frequency. It might be expected that the Carrier characteristic itself is lower, which is the largest distance between any of the listed feature distributions. However, this effect extends to lowering the distance for most other characteristics, indicating that using a representative control is worthwhile in qualifying the faults in a sample.

While controlling for Carrier EARFCN 3510 does seem to influence the distribution distances of many features values, the effect appears to be largely independent of SINR, TA, and PRB Usage. When performing fault analysis, a telecommunications expert with an understanding of network interacts will be able to select appropriate control filters as part of the analysis.

Table 6: Wasserstein distance between SHAP distribution in a problematic sample for Cell *217442-10* and a representative baseline (CDRs with Carrier EARFCN 3510) or a randomized baseline (CDRs from all frequencies)

Statistic	Cell 217442-10	Representative Baseline	Random Baseline
Mean	1935.609	3301.991	6902.786
Standard deviation	1564.512	3625.342	8973.761
Sample size	2868	10000	10000

	Baseline	
Feature	Representative (w_1)	Random (w_1)
Roamer	**77.652**	165.679
EnodeB	**378.111**	419.644
Reference Signal Received Quality (RSRQ)	**102.119**	187.344
Reference Signal Received Power (RSRP)	**117.914**	210.279
Signal-to-noise Ratio (SINR)	51.429	50.324
Quality of Service Class Identifier (QCI)	**97.572**	204.877
Carrier Frequency	**255.961**	897.025
Service Used	**54.534**	101.820
Timing Advance to Cell	320.074	337.632
Cell	**181.521**	380.454
Cell physical resource block (PRB) Usage Ratio	161.431	152.632
Handset Manufacturer	**173.276**	220.402
Handset Type	**83.277**	100.094
Mean	**97.973**	163.384
SD	**111.990**	214.762
Total distance	**2057.430**	3431.062

Table 7 extends this investigation to samples of the 10 lowest download average throughput cells with Carrier EARFCN 3510 in the dataset. The w_1 feature distances between samples from these problematic cells and representative or random controls are summarized in the table. Again, the table demonstrates consistency lower *mean* and *total* w_1 for the representative baseline that controls for the frequency. This indicates that controlling for certain characteristics will produce more representative baselines and, thus, be valuable as part of a fault analysis.

These results also suggest the model is complex enough to represent complex relationships in the dataset, allowing it to distinguish between distinct subset characteristics in the data without necessarily having to form model subsets separately. This point is critical for a practical implementation of a fault analysis strategy because, without it, specific questions usually posed in analysis would have to be known a priori (i.e., before forming a model) and could not be asked by a team of telecommunications experts while using an interactive analysis tool. While the results are not included here, separate experiments confirmed that models formed on the subset of frequency-specific CDRs only offered very minor improvement in throughput prediction relative to models not restricting frequencies.

The implications for a fault analysis system are demonstrated in Table 8, which shows the differences in w_1 distances and median differences for Shapley value distributions between samples from problematic Cell 217426-10 and controls. Again, w_1 distance between the Shapley values provides the distances between the mass of the PDFs of the compared samples distributions (i.e. Cell 217426-10 compared to a control sample). The median difference provides a non-parametric way of determining the direction of the distance by its sign.

Using the distribution distance w_1 in Table 8, a rudimentary ranking procedure for fault analysis becomes clear. Ranking features based on their SHAP distribution distance from controls provides an order of relevance in association to degradation on the outcome we care about. As might be expected, there are different feature rankings for each control. The goal in this fault analysis is to uncover possible contributing factors to an observed problem, therefore characteristics with positive Shapley values (i.e., expected by the model to *improve* throughput relative to the mean) are ignored. Taking the top-5 feature level characteristics in order of distance from what was expected, the random control would indicate that Carrier Frequency, EnodeB, RSRQ, RSRP, Cell warrant further investigation for perceived contributions to poor throughput. However, using a more representative baseline that controls for Carrier EARFCN 3510, a more appropriate story emerges where the EnodeB, RSRQ, Cell PRB Usage Ratio, TA to Cell and Cell appear to be the most relevant features that warrant further investigation. Incorporating this choice into a fault analysis tool would allow experts to dynamically chose appropriate controls when evaluating

Table 7: Wasserstein distance between SHAP distributions for 10 lowest mean throughput cells on Carrier EARFCN 3510 and control samples

Cell	Download Average Throughput (kbps)		
	Sample (n)	Mean	SD
217426-10	1391	1267.006	1129.745
218871-9	1134	1504.011	1631.577
217058-10	1214	1562.401	1245.212
217904-10	862	1575.892	2842.743
217058-9	1278	1592.791	1286.613
217356-9	962	1609.179	2487.040
217991-10	1857	1627.012	3171.052
217170-9	749	1688.593	2122.803
218453-9	1473	1790.895	1747.628
218021-9	1131	1837.066	1421.247

Cell	Representative Control w_1			Random Control w_1		
	Mean	SD	Total	Mean	SD	Total
217426-10	**98.215**	118.403	**2062.517**	200.679	260.230	4214.265
218871-9	**89.049**	122.919	**1870.033**	192.216	298.296	4036.549
217058-10	**116.213**	160.652	**2440.490**	187.373	248.120	3934.848
217904-10	**117.542**	204.724	**2468.383**	224.391	378.519	4712.227
217058-9	**103.828**	151.288	**2180.391**	179.744	256.927	3774.624
217356-9	**95.019**	124.534	**1995.419**	203.072	332.545	4264.531
217991-10	**101.525**	133.971	**2132.031**	211.332	339.262	4437.988
217170-9	**82.856**	118.805	**1739.994**	185.938	296.315	3904.702
218453-9	**84.823**	129.972	**1781.290**	177.880	262.348	3735.489
218021-9	**78.265**	122.045	**1643.572**	180.678	265.255	3794.258

samples.

The most interesting difference in the feature rankings is that the RSRP is much less problematic when compared against the representative sample than a random sample, taking RSRP out of contention as a problematic feature. This demonstrates the value of domain expertise in the application of this fault analysis system, as RSRP is inherently tied to Carrier frequency characteristics and controlling for one will impact the relevance of the other. Importantly, the model representation included these relationships, otherwise drawing different samples would not have produced different feature importances. Domain knowledge was required to apply the appropriate control, however, so characteristic selection for the control would have to be a step of the analysis system.

As part of an automated system it would be possible to flag minimum distances for further exploration and, eventually, apply a new layer of rule-based systems to automate some of the evaluation process. For example, given a threshold distance of $w_1 > 300$, EnodeB, RSRQ, Cell PRB Usage Ratio might suggest congestion on the Cell and EnodeB affecting service quality and throughput. These results also indicate that, in general, this approach for fault analysis offers an automation layer to learn relations in the data automatically and *not* an avenue for full automation.

4.2.3 Stability of Representation

One difficulty in using complex datasets is the ubiquity of correlated input features. Many statistical strategies assume independence of features prior to modelling, but this assumption is commonly broken. Stability is a characteristic described in computational learning theory which, if present, indicates that minor changes in the data characteristics such as the ordering used to train a model or the subset of data chosen to represent the data, will not considerably change predictions. Thus, models produced through a stable learning algorithm trained a on a largely similar data should be largely the same.

When using the model for its representation instead of its prediction, the stability of representation is also important. A model produced through a gradient learning algorithm will be different based on the order of presentation of the data, particularly when the input features are correlated.

This section will serve to validate that the learning approach used and the SHAP evaluation of fault samples is stable. In practical terms, how consistent is the story told by models trained on differing data? We compare representation outcomes between different models trained on subsets of the data (i.e., folds) and the ordered presentation to the learning algorithm is different. For each fold, stability was evaluated by comparing the SHAP representation for each model on cases

Table 8: Wasserstein distance of SHAP distributions between Cell *217426-10* and controls as might be used in feature ranking and selection

	Representative Ctrl.		Random Ctrl.	
Feature	w_1 Dist.	Median Diff.	w_1 Dist.	Median Diff.
Roamer	77.538	13.250	166.781	44.495
EnodeB	**364.749**	-364.177	**436.047**	-392.454
Reference Signal Received Quality (RSRQ)	**321.219**	-301.447	**407.840**	-391.270
Reference Signal Received Power (RSRP)	143.903	-265.029	**381.941**	-539.685
Signal-to-noise Ratio (SINR)	55.123	-24.458	119.992	-67.308
Quality of Service Class Identifier (QCI)	58.045	28.712	211.230	57.059
Carrier Frequency	84.419	24.158	**1116.70**	-1100.733
Service Used	57.084	-29.855	152.966	-58.159
Timing Advance to Cell	**187.889**	-157.219	230.867	-145.295
Cell	**178.584**	-126.833	**374.237**	-223.467
Cell PRB Usage Ratio	**342.614**	-360.928	314.677	-345.388
Handset Manufacturer	117.434	-20.259	192.370	9.966
Handset Type	69.553	-8.948	105.381	-6.240

in a held-out test set.

4.2.3.1 Experiment: SHAP Stability

Five models were created, using a standard k-fold cross-validation approach but keeping each of the 5 trained models to evaluate against test data. In order to form an appropriate sample for evaluation, samples of CDRs were drawn based on their connection to a particular EnodeB, a hardware component in the RAN network. To make the samples pertinent for a fault analysis use case, the selected EnodeB were chosen on the basis of association with the lowest average download throughput CDRs *and* their being in the top 20% most frequently used EnodeB; these constraints aim to assess severity and impact, respectively. By this criteria, the following 5 EnodeB were evaluated: *217339*, *217937*, *218127*, *218426*, *227657*.

The results of the first stability results are shown for a specific EnodeB *217339* in Table 9. This table displays the average SHAP contributions per feature group as determined by the model fold for the *same* 10,000 CDRs in the problematic EnodeB *217339*. Because sample selection was determined based on the worst performing EnodeB, the lower expected throughput for the EnodeB feature is a good indicator that the model interpretability strategy is working in attributing feature importances (i.e., we expect that this EnodeB is problematic). In taking the mean result for each feature across the sample, as in Algorithm 1 without reordering, the estimated contribution of each feature is consistent for each trained model.

Some variability is to be expected because each trained model has only seen $4/5^{th}$ of the possible data in the 5-folds. However, these results show a high stability of representation based on the low variation between model folds with respect to the average feature contributions to the target prediction (e.g., throughput) for each data line (e.g., CDR) and the same direction in the importance for those values.

The CDR level analysis, shown in Table 10, compares Shapley values produced by each model fold on a *case-by-case* basis for the same 10,000 unseen CDRs of EnodeB *217339*. This table gives better visibility on the variability between folds by evaluating variability *per CDR*. This evaluation is important as the aggregations presented in Table 9 are using an averages of Shapley values per feature for the entire model fold, which might hide variations between the models for individual CDRs. This evaluation is particularly important for categorical variables as they unevenly occur in a CDR sample (e.g., few occurrences of a particular handset).

To interpret Table 10: given an *individual CDR*, this table shows the mean, standard deviation between folds, coefficient of variation (as measured by the standard deviation over the mean) and the range (as measured by the absolute difference between the highest and lowest Shapley

Table 9: Mean SHAP by model fold on EnodeB *217339*

Feature	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Roamer	23.59	24.24	24.62	24.11	25.22
EnodeB	-801.73	-785.74	-774.84	-789.93	-772.96
Reference Signal Received Quality (RSRQ)	-12.06	-11.83	-11.67	-14.97	-12.53
Reference Signal Received Power (RSRP)	-62.21	-65.88	-64.47	-65.53	-65.24
Signal-to-noise Ratio (SINR)	18.89	17.77	23.71	12.74	16.71
Quality of Service Class Identifier (QCI)	-88.34	-87.25	-91.76	-90.63	-93.47
Carrier Frequency	58.70	57.68	56.50	55.23	57.89
Service Used	-64.52	-67.55	-63.28	-58.87	-64.47
Timing Advance to Cell	-182.59	-187.25	-185.94	-185.12	-187.19
Cell	14.38	8.82	20.08	17.31	9.04
Cell physical resource block (PRB) Usage Ratio	-271.13	-274.10	-274.20	-250.38	-264.71
Handset Manufacturer	-97.42	-93.71	-97.78	-95.03	-98.59
Handset Type	-32.49	-31.12	-30.86	-29.00	-26.23

values) across the 5 fold models for that CDR. A good indicator for the stability of representation is the average standard deviation between folds, with a lower value representing a more stable representation for this feature.

The variability in feature contributions range from *5.21* to *29.42*. In general, the size of the SD is proportional to the mean SHAP of any particular feature on the prediction. This proportion is measured by the average coefficient of variation, which is the average $sd/|mean|$ SHAP per CDR across folds. A high average coefficient of variation may indicate possible stability issues with categorical variables such as Cell and Handset Type. This instability is possibly due to there being fewer training examples for each of the categories and, thus, an uneven exposure based on fold divisions. Accounting for infrequent categories may be required to ensure stability for category variables. Another plausible explanation is that small amount of variability is inherent to the training process. This could mean that a high coefficient of variation is an artifact of *low* feature importance and *constant* variability between the model folds because of a baseline instability. Further experiments will be required to rule this out.

To demonstrate stability in different samples, Table 11 extends the model fold comparison to the 5 worst EnodeB, consisting of 10,000 CDRs each and selected based on the severity and impact factors for Download Average Throughput described previously. This demonstrates the same

Table 10: Variability in SHAP *per CDR* across model folds on EnodeB 217339

Feature	Mean	Per CDR Average SD b/w Folds	Per CDR Average Coefficient of Variation	Per CDR Average Range b/w Folds
Roamer	24.35	5.21	0.04	12.94
EnodeB	-785.04	29.42	0.04	72.84
Reference Signal Received Quality (RSRQ)	-12.61	15.47	0.68	38.26
Reference Signal Received Power (RSRP)	-64.66	16.67	0.64	41.25
Signal-to-noise Ratio (SINR)	17.96	14.29	0.80	35.48
Quality of Service Class Identifier (QCI)	-90.29	12.44	0.64	30.64
Service Used	-63.74	14.67	1.04	36.26
Timing Advance to Cell	-185.62	20.38	0.14	50.56
Cell	13.93	23.50	2.02	57.81
Carrier Frequency	57.20	9.28	0.57	22.91
Cell PRB Usage Ratio	-266.90	23.85	0.71	59.19
Handset Manufacturer	-96.51	17.09	0.19	42.18
Handset Type	-29.94	19.14	4.81	46.67

stability results shown in 9 for an arbitrary selection of features (continuous and categorical), to show stability within samples and differences between samples, as these have distinguishing characteristics. For example, SINR for EnodeB *227657* is evaluated as problematic for each fold model, whereas the SINR in EnodeB *218127* has a positive influence on the throughput prediction. In most cases, the variability between the model fold characterizations, as measured by the standard deviation, are considerably smaller than the mean characterization differences for the samples. Thus, the stability for this strategy is sufficient to make meaningful comparisons between samples that are not reducible to noise in the sampling or training process.

While these tests serve to ensure stability in the learned representation and interpretation with SHAP, the ultimate goal of the fault analysis strategy is to characterize full samples. Table 11 establishes that the complement of feature values in each sample are quantitatively different based on the weighted SHAP importances in a way that is consistently interpreted to be positive or negative by the five different models. This consistency strengthens the idea that, given the type and quantity of data made available for this use case, the model representation is sufficiently stable to support an analysis framework.

Although the variability between model folds is small, this variability may be caused by the high degree of correlation between input features as these can be used interchangeably during the model training process. For instance, consider that Handset Type is a subset of manufacturer and end_cell is a subset of EnodeB, and, thus, are perfectly correlated. Moreover, other network metrics included here describe qualities of the network signal that are often positively or negatively correlated. As the input features are not independent, it is expected that the learning algorithm will use different characteristics from data in the model formation process. However, even if such correlations increase instability in the representation, these experiments suggest that they will not hamper the utility of the approach for fault analysis in this dataset.

4.2.4 Analytical Exploration vs. Statistical Understanding

An insightful framing of analytical vs. statistical strategies is presented by Cassie Kozyrkov, Chief Decision Scientist at Google [7]. In this framing, analytics offers a description of observed data, which can provide insight and open up avenues for exploration. Once this insight is obtained, statistics is the process used to draw conclusions about data to generalize them to data not yet been seen. While either an analytical or statistical framing would be valuable for this project, the primary goal for this fault analysis system is analysis; the system aims to facilitate the exploration of faults to allow domain experts to follow up with concrete investigations. This section evaluates

Table 11: Stability comparison across model folds for distinct samples

EnodeB	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	SD
EnodeB Mean SHAP							
217339	-801.73	-785.74	-774.84	-789.93	-772.96	-785.04	11.76
217937	-1043.78	-1066.25	-1052.20	-1068.59	-1061.74	-1058.51	10.35
218127	-1304.55	-1349.48	-1353.95	-1320.52	-1321.69	-1330.04	20.97
218426	-1333.31	-1332.82	-1341.64	-1335.26	-1316.06	-1331.82	9.49
227657	-820.33	-826.36	-834.51	-831.66	-870.15	-836.60	19.52
Signal-to-noise Ratio (SINR) Mean SHAP							
217339	18.89	17.77	23.71	12.74	16.71	17.96	3.96
217937	-9.31	-7.84	-10.30	-9.07	-9.68	-9.24	0.91
218127	47.63	41.67	47.87	40.48	41.79	43.89	3.56
218426	-1.48	-2.42	-1.84	-2.96	0.01	-1.74	1.13
227657	-95.18	-86.64	-83.96	-84.31	-86.32	-87.28	4.57
Timing Advance to Cell Mean SHAP							
217339	-182.59	-187.25	-185.94	-185.12	-187.19	-185.62	1.92
217937	-18.82	-17.30	-18.25	-23.54	-21.03	-19.79	2.51
218127	-57.05	-56.68	-53.67	-59.60	-57.60	-56.92	2.14
218426	-23.90	-28.49	-29.21	-29.23	-31.91	-28.55	2.91
227657	-139.63	-139.50	-138.65	-130.22	-123.56	-134.31	7.18
Handset Manufacturer Mean SHAP							
217339	-97.42	-93.71	-97.78	-95.03	-98.59	-96.51	2.05
217937	-84.07	-86.20	-80.63	-85.93	-84.29	-84.23	2.22
218127	-104.22	-105.92	-108.54	-106.85	-111.63	-107.43	2.82
218426	-125.14	-125.22	-128.18	-120.87	-132.39	-126.36	4.26
227657	-68.00	-71.80	-69.68	-68.97	-75.49	-70.79	2.98

the implications for this approach.

4.2.4.1 Implications for Fault Analysis Strategy

Performing a model evaluation step on unseen data is the standard for machine learning, as the goal is typically to create a model to perform predictions on new data. This step serves to ensure the model has learned something fundamental about the true data distribution of the data; that the model generalizes and does not simply memorize the observed data points. However, because fault analysis of a sample can involve using a model to aggregate or summarize observed data for interpretation, this validation step may not always be necessary; the model becomes a summarization of the data it has seen rather than a representation that generalizes to unseen data. Consequently, we have considered two possible approaches:

One approach evaluates the model against observed data while being reasonable in the constraining of the model size; there cannot be as many model parameters as datapoints because the model will memorize the training data and, thus, does not summarize the data in any useful way. As a preventative measure against overfitting, and initial model training could hold out data for training (e.g., 20%) and be trained until validation error stops improving or diverges to establish an appropriate number of training epochs at which to stop training the model. Once appropriate hyperparameters that optimally reduce validation loss are selected, all available data are included in a subsequent model on which the model interpretation algorithm is applied for analysis. This strategy is more time consuming, as it involves the subsequent creation of two models, but ensures that all available data was used in the analysis while also providing a modest check against overfitting. This strategy uses the model includes all the data that will be evaluated using SHAP.

Alternatively, a second approach is to *only* evaluate the model performance and interpretation against unseen data. This approach aligns well with machine learning best practices, but other considerations arise when taking the approach. For instance, verifying the quality of a particular prediction prior to evaluating model feature attributions for that prediction becomes very important when there is no guarantee that the model has had any exposure during training to such an example. Because the model has not incorporated information from the particular cases being evaluated, it weakens the case for the analytical discovery of specific patterns in the data that could be meaningful, but transient. However, this approach increases our ability to draw generalizable conclusions from the analysis.

4.2.4.2 Experiment: Seen vs. Unseen Data

This experiment evaluates the performance of the model in evaluating data seen and unseen by the model. Sample selection was done by first finding the lowest median throughput EnodeB that was also in the top 20% highest volumes with sufficient sample size and impact. These CDR samples were thus drawn from the same problematic EnodeB used in the stability section: *217339*, *217937*, *218127*, *218426*, *227657*.

CDRs are call traces which means that several CDRs can characterize a user interactions with the telecommunications network over multiple transactions. Because we are not using sensitive identifying information that would allow us to distinguish between individual users, a random sample of CDRs across the dataset will likely contain CDRs by the same users and, worse, the same transaction with the network. To mitigate the risk of drawing CDRs by the same user for the same transaction in both the *seen* or *unseen*, the dataset was split in half, using the first half to train the model and second half for unseen samples. Assignment to *seen* or *unseen* groups were made at random for each EnodeB, drawing from the first or second half of the dataset, respectively.

While the throughput results are somewhat regular with large sample sizes, the high skew make the median statistic a more appropriate measure in evaluating throughput prediction error. There are occasional, high throughput cases that lead to very large prediction errors in the model, which can be observed in Figures 8 and 9. As would be expected, the median line in either figure can be found at the center of the probability mass, although a higher percentage of the errors values in the histogram cluster toward lower error rates.

Table 12 demonstrates prediction errors in throughput when evaluating *seen* and *unseen* data for the five selected EnodeBs. The absolute median model errors for all samples are consistently lower for seen data, which suggests the model is better representing these cases. When instead looking at the median error (i.e., not *absolute* median error) the median values are centered at roughly *0* for of the *seen data*, indicating an equal proportion of over-prediction and under-prediction. This may indicate that prediction errors are attributable to noise and contributions from other, unknown variables such as the service being used by the end-user. In this example for instance, CDR selection for each EnodeB in the unseen data were made at a different time of day, which may have changed usage patterns and the state of the telecommunications network resulting in the differences observed for unseen data.

Another possible implication of this prediction error is that the unseen data is out of distribution, which again raises the question as to whether the aim in fault analysis system is to describe the relationships in the observed in the data through analysis or extend conclusions to

the network more generally. Regardless, the median prediction error has implications for the fault analysis system because these predictions values will be fairly distributed between input features as Shapley values. If there is a systematic over-prediction or under-prediction, this will also affect the attributed importances of specific features. Thus, while the model behavior can be explained through SHAP for seen or unseen data, the *data* is more appropriately explained by SHAP for seen data; the fault analysis system aims to explain the *data*, not the model.

Table 12: Wasserstein distance between Shapley values generated on seen and unseen data

			Avg. Throughput (kbps)			Prediction Error	
		Size	Median	Mean	SD	Median	Median Abs.
217339	Seen Data	10000	2801.646	4426.528	5206.050	-4.117	1173.358
	Unseen Data	10000	2389.467	3521.754	3787.199	-559.602	1591.118
217937	Seen Data	10000	2324.686	3749.688	4656.388	-2.191	789.902
	Unseen Data	10000	2593.403	4575.861	5866.191	85.078	1372.485
218127	Seen Data	10000	2288.000	3894.968	5210.933	0.559	1058.532
	Unseen Data	10000	2083.104	3317.948	4019.789	-274.830	1341.896
218426	Seen Data	10000	1789.242	3545.208	5393.790	-0.086	532.807
	Unseen Data	3048	1962.203	4130.702	6303.293	-70.976	1065.058
227657	Seen Data	10000	2223.272	3664.748	4726.869	0.456	782.570
	Unseen Data	10000	2317.472	3387.768	3771.313	-345.780	1281.227

4.3 Visualization Strategies

A fault analysis system has to include both localization and qualification of the problem. Because the described strategy aims to model and represent complex relationships in the data, presenting the right amount of detail is key in making a useful application for fault analysis.

The SHAP visualization library is well developed, with many options. For example, there are SHAP plot summaries that give visibility overall feature importance of a sample, feature attributions for an individual example (i.e., force plot) and partial dependence, to name a few [11]. These graphs provide a lot of detail and include best-practices in explainability in that they show point-cloud distributions that aid the viewer in developing an intuition for the results. However,

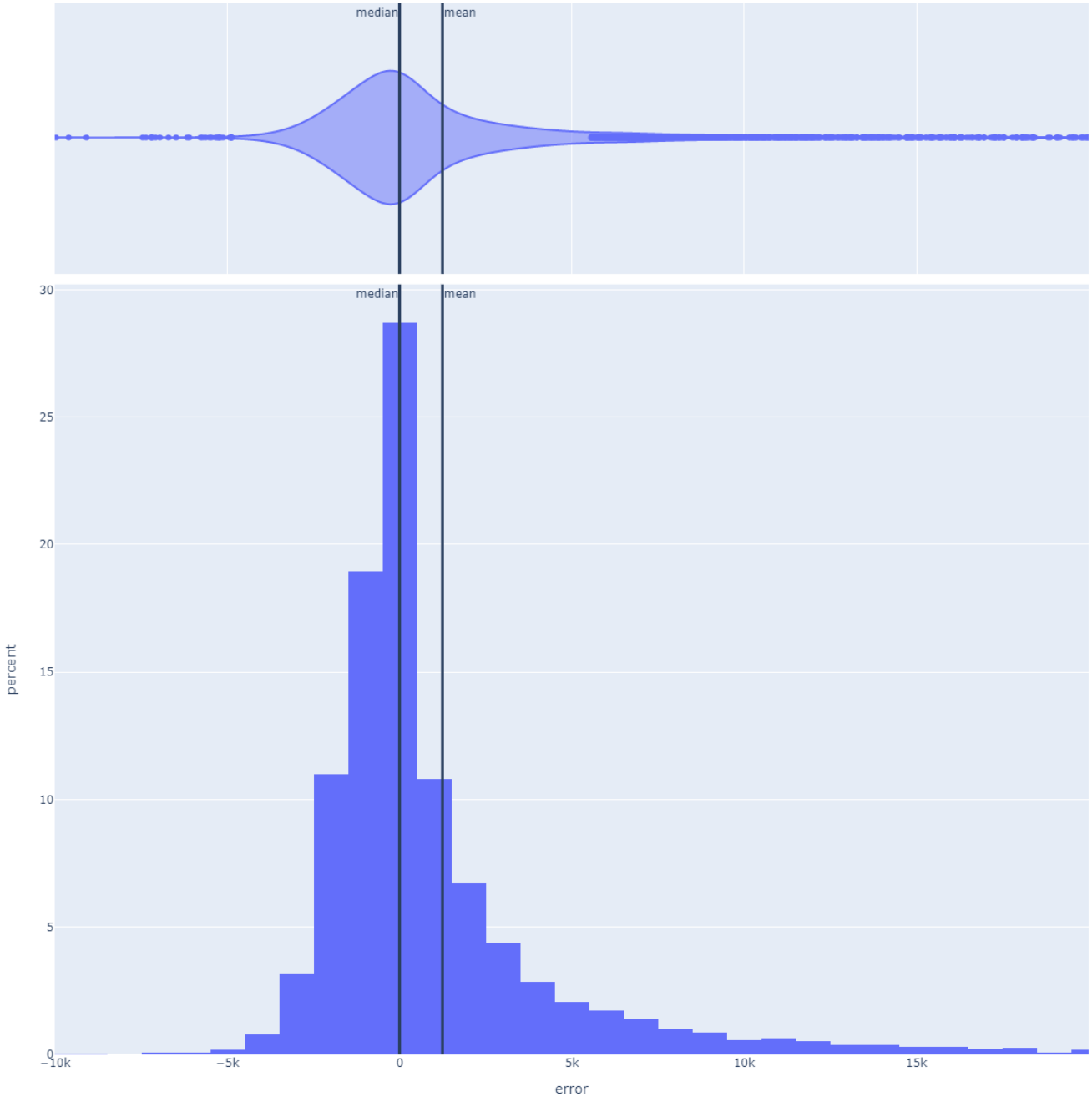

Figure 8: Model error violin (top) and bar (bottom) density plots of seen data for EnodeB *217339*

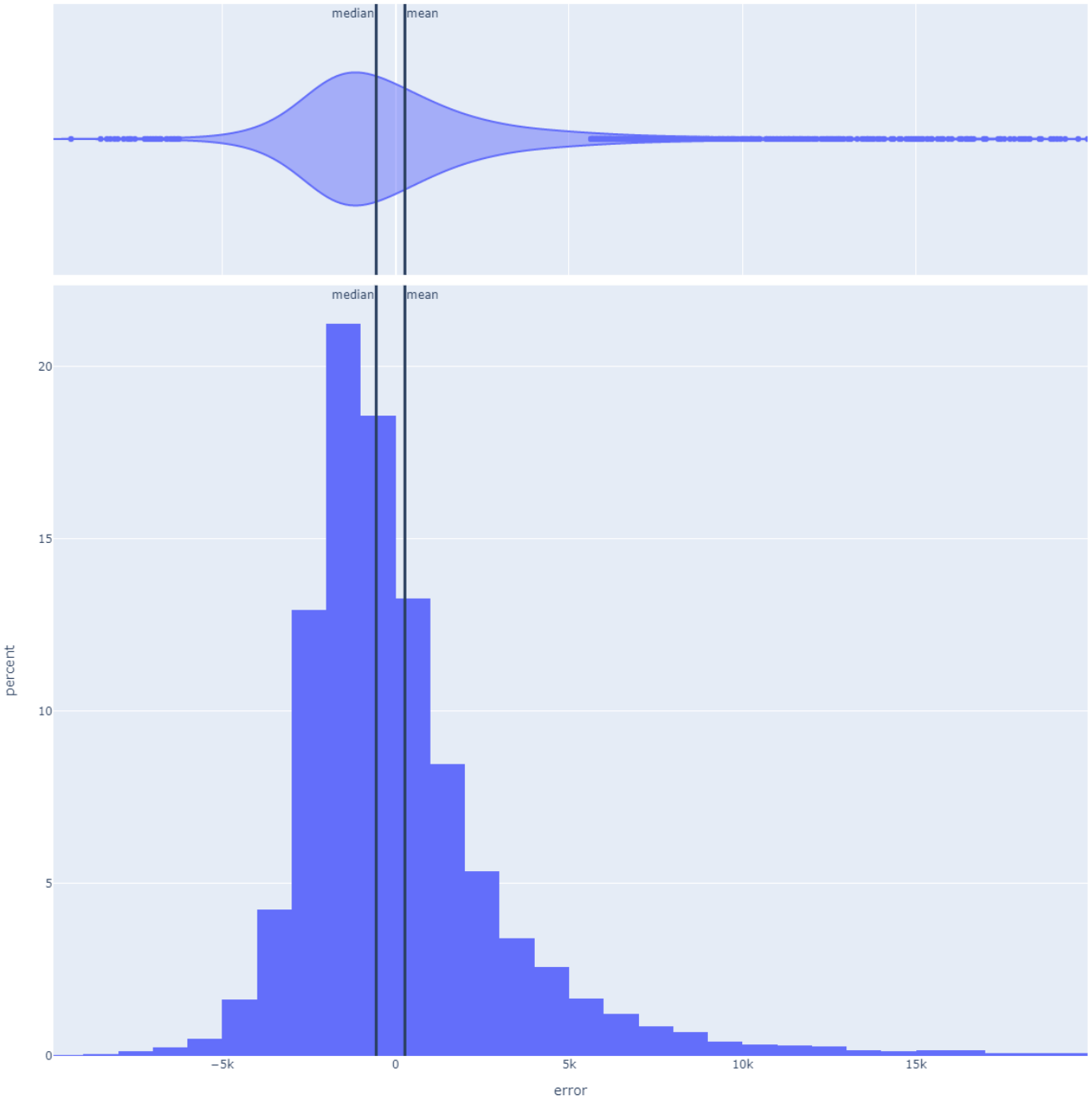

Figure 9: Model error violin (top) and bar (bottom) density plots of unseen data for EnodeB *217339*

while these plots offer an appropriate amount of detail for a user who is comfortable with SHAP, their complexity and lack of directness may make it difficult for fault analysis by those responsible for monitoring information systems.

We developed a visualization tool that highlights the unusual characteristics found for problematic samples where faults are expected to have occurred. These visualization are based on a severity and impact concept; where severity represents how "bad" the characteristic was when it did occur, and impact distributes this severity based on the number of occurrences in the sample. Severity is determined based on the SHAP aggregation Algorithms 1 and 2. Impact is a measure used to represent the overall association of a characteristic with degradation (e.g., lower throughput) in a sample. Impact is calculated by multiplying the severity of the characteristic by the frequency of the occurrence in the sample.

Using severity and impact is particularly useful in bringing out the relative contributions feature groups (e.g., an problematic categorical column) and the contributing values (e.g., a problematic category) when a particular model was trained with categorical data. While this visualization will not be as nuanced or complete as point clouds visualization included in the open-source SHAP project, it may be more useful for fault analysis when the principles of machine learning and SHAP are not an area of expertise, such as is the case for many telecommunications professionals.

The following visualizations demonstrate results for a sample from a geographical region of interest *Polygon 5*. This area was selected by telecommunications professionals on the basis of it being a problematic area of the RAN network

4.3.1 Contributions highlights

To quickly display characteristics with the highest contributions to *lowering* throughput for the sample, a highlights table is displayed. The highlighted features tables shows the most impacting feature-value pairs in the categorical case and quantile ranges in the continuous case in their order of impact on the lowered performance for the full sample. Severity score is calculated by Algorithm 2. Multiplying the *severity* by the *fraction* of affected cases in the sample provides an *impact* score used to represent the most important characteristic related to degradation for the sample.

In the example shown in Figure 10, the characteristic with the highest overall impact in *lowering* expected throughput was feature EnodeB 217287. The impact score of 444.42 indicates an average reduction in throughput of 444.42 kbps from the feature EnodeB 217287. The fraction column indicates that 52% of the CDRs in the sample are on EnodeB 217287. Finally, the severity score indicates the average SHAP calculated for the characteristic (using Algorithm 2). Thus, according

to the model, EnodeB 217287 accounted for an average degradation of 854.65 kbps when it was a characteristic of the CDR in this sample.

The highlight table concept can also include sample level statistics related to each characteristic which may be unique to the information systems domain. For example, later iterations of this highlights table included the number of affected subscribers with this characteristic and uniformity of the CDRs these users accounted for. This additional information can aid in an assessment of the results by helping domain experts draw conclusions with information they do not want to explicitly include in the model representation.

Highlights

	element	impact	severity	fraction
×	end_cell_id_enodeb: 217287.0	444.42	854.65	0.52
×	end_cell_id_enodeb: 217968.0	160.07	666.96	0.24
×	detailed_service_id: terminating-LTE-data-call	159.61	693.96	0.23
×	end_cell: 217287-14	152.14	422.61	0.36
×	last_rsrp-q1: (-120.5, -106.5]	125.17	625.85	0.2

Figure 10: Highlighted characteristics in order of contribution impact for *Polygon 5*

4.3.2 Contributions Flow-diagrams

Flow diagrams are used to represent the contributions of features and specific feature value bins to the degradation in a problematic sample. The intention of these diagrams was to provide a visual aid to rank the contributions, making it easier to evaluate relative importances for follow-up investigation. The impact score described in the previous *contribution highlights* section is used to represent the size of the diagram nodes. The nodes colored in blue represent dimensions (i.e., categorical values) and orange to represent metrics (i.e., continuous values) as the telecommunication experts indicated this will facilitate analysis for them. Because of our focus on fault analysis, only those features with sufficiently negative associations to throughput in the sample are included in the visualization, which is governed by an importance threshold (e.g., average SHAP feature contribution ¡ -100 kbps).

In the flow-diagram shown in Figure 11, a Sankey Flow Diagram [4] is used to show the proportion of each feature contribution to the throughput reduction of 30.18% in the *Polygon 5* area. The size of each node is represented by the impact score. The severity used to determine impact in this case is the SHAP value computed in Algorithm 1.

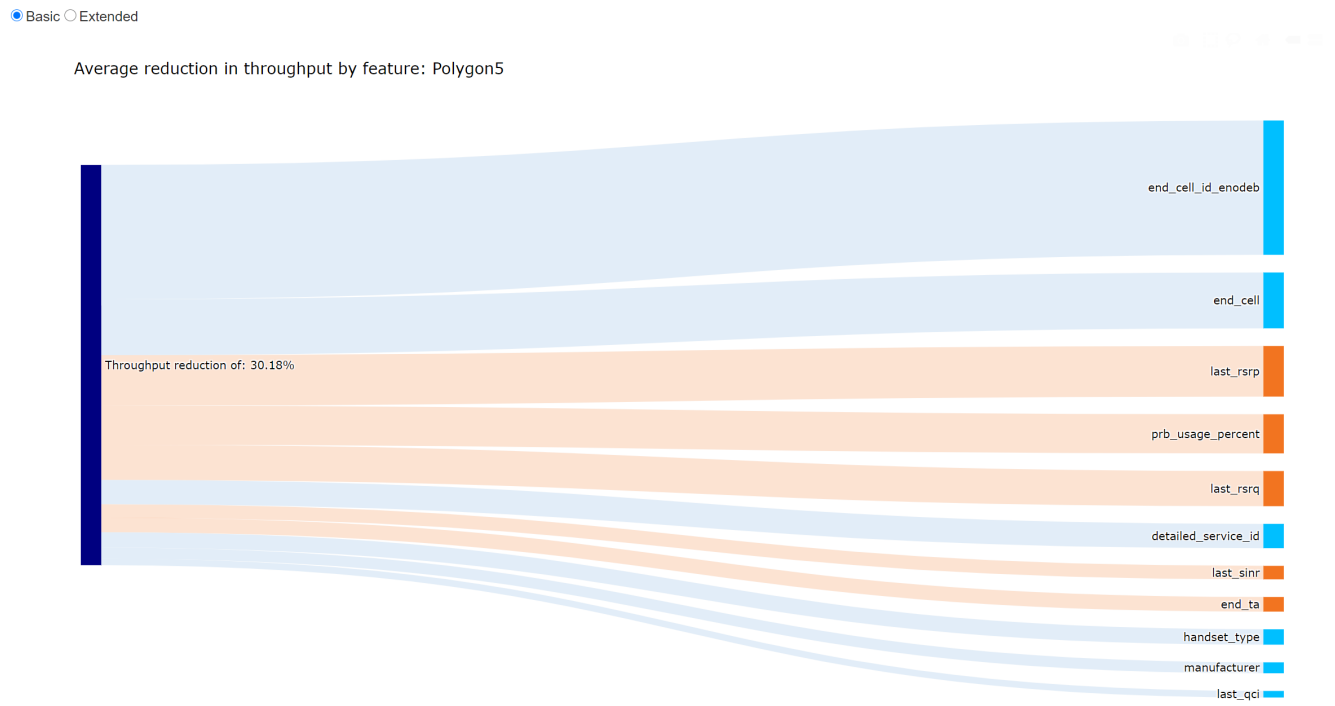

Figure 11: Features contributions visualization for *Polygon 5*

The flow-diagram shown in Figure 12 is an extended feature view with the contributions of feature value bins in the selected region *Polygon 5*. While the characteristics are largely the same as the feature view, a top-k selection of highest impact categorical instances in the sample are shown for each problematic dimension (e.g., manufacturer) and the problematic 5-quantiles ranges as determined by q-cut binning are selected for each problematic metric. The impact score are determined by calculating SHAP severity by Algorithm 2. 5-quantiles were chosen for continuous metrics because this was the quantile range typically by telecommunication experts to perform manual analysis, but this can be modified to suit the domain.

4.3.3 Confidence Bounding

Figure 13 shows an application of the scaled error heuristic introduced in Chapter 3 to create confidence bounds on the results of *Polygon 5*. The figure shows a histogram with ordered presentation of the impact of features-value in addition to the proposed confidence bounding heuristic. A *higher* impact for a characteristic had a *lower* SHAP value and is interpreted as *reducing* the expected throughput.

The confidence bounds provide visibility on whether the model was over-predicting or under-predicting in the presence of the particular characteristic. In Figure 13, these bounds are displayed

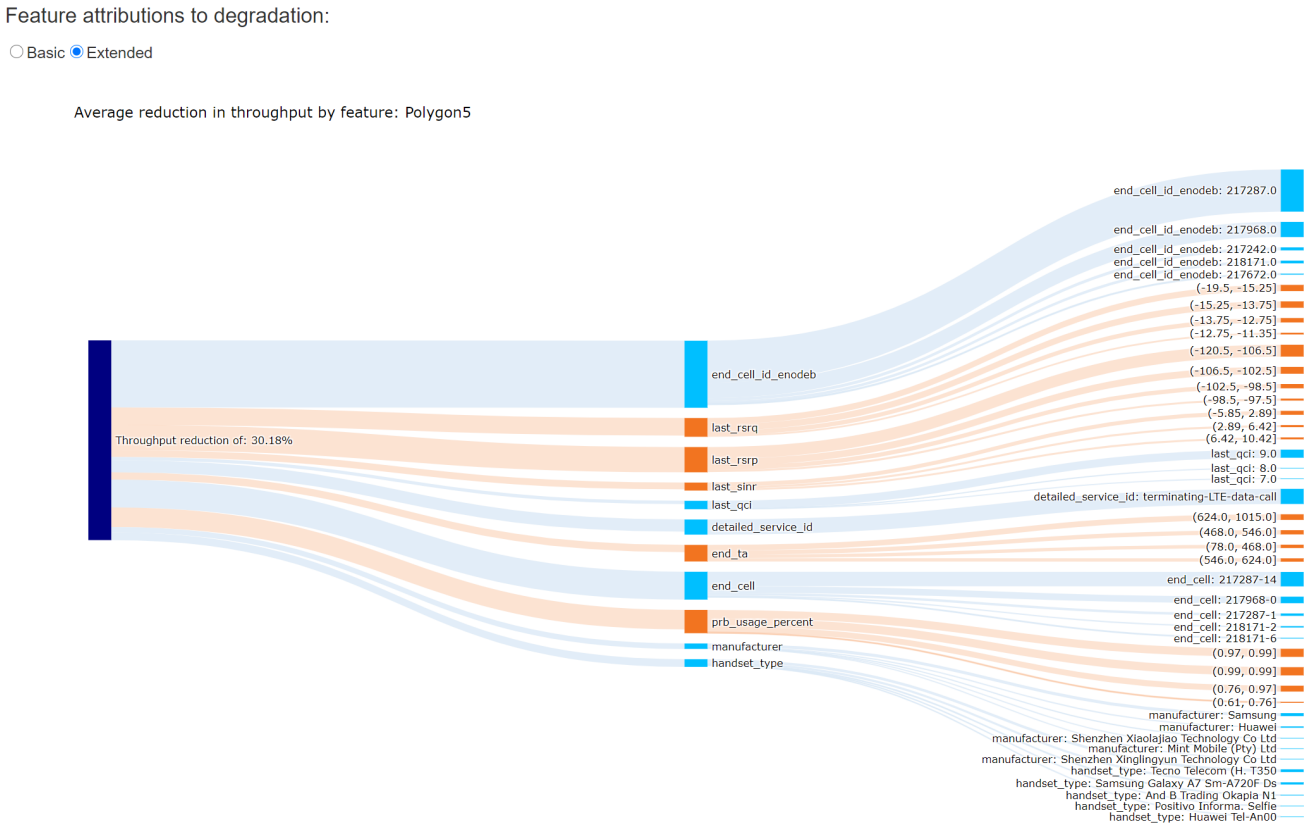

Figure 12: Features and feature-value set contributions visualization for *Polygon 5*

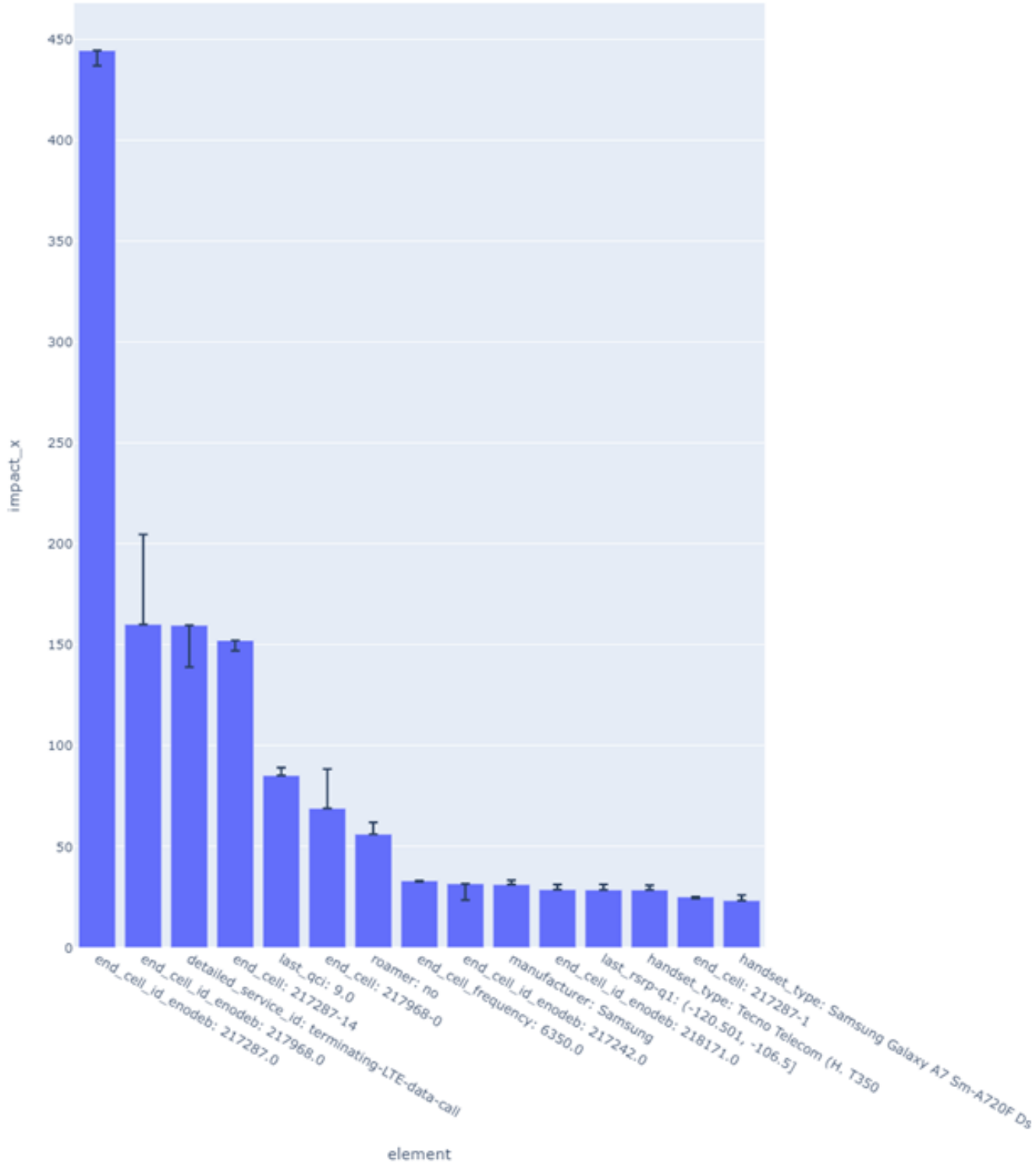

Figure 13: Feature impact with confidence bounds based on scaled prediction error for *Polygon 5*

as error bars, but are an entirely different concept than traditional error bars. The intuition in interpreting these bars is the following:

- Predictions for CDRs with the characteristic *end_cell_id_enodeb: 217968.0* were systematically *higher* than they should have been (i.e., CDRs with that characteristic were *worse* than the model expected them to be)
- Predictions for CDRs with the characteristic *detailed_service_id: terminating-LTE-data-call* were systematically *lower* than they should have been (i.e., CDRs with that characteristic were *better* than the model expected them to be)
- Predictions for CDRs with *end_cell_frequency: 6350.0* were *neither* systematically higher nor lower, or there was no prediction error; this case is most likely to be attributable to noise and is the best outcome for the model

Another useful addition to this heuristic might be to include the variability of the bounding values. While these are currently one-sided as they represent the average calculate bound, it could be useful to know how high the error was even if the model was not systematically over-predicting or under-predicting.

Although the confidence bounding heuristic is not well grounded theoretically, this style of feedback may be helpful in preventing experts from drawing incorrect conclusions when the model has a poor representation of local samples with specific characteristics.

4.4 Validation on Known Relationships

A qualitative assessment of the results on this dataset demonstrated that the model does capture known relationships between the input values and throughput. Often, the telecommunications experts collaborating with us on this project had intuitions about which values represented poor quality metrics based on repeated exposures to these concepts in their work. In general, the observed relationships in different problematic samples provided some insight into characterizing what had gone wrong.

Critically, although these relationship between metrics and Download Average Throughput have been confirmed by domain experts, they were learned from the data and did not require any injection of domain knowledge. It is our belief that verifying these learned relationship with domain experts will be necessary to validate any new system. Nevertheless, having a system that learns

and quantify relationships from the data without intervention will greatly improve our ability to perform analysis of faults in complex systems.

The following examples demonstrate the learned relationships between continuous feature values and download average throughput in the dataset. This following graphs were produced from the standard SHAP visualization library. The Shapley values displayed were estimated for a randomly drawn sample of CDRs in the dataset:

Figure 14 shows an exponential relationship between Download Average Throughput and RSRQ (i.e., the Shapley value). The direction of this relationship is intuitive in that we expected signal quality would be directly correlated with throughput for a downlink connection. However, the quantified attribution to a specific amount of throughput for a particular metric value would be difficult without this fault analysis approach using SHAP. This dependence plot shows an exponential increase in the expected throughput when moving from a low (i.e., -20) to high (i.e., -2.5) RSRQ. Further, the variability in Shapley values for different CDRs of the same RSRQ indicate that changing contexts will increase or decrease the impact of RSRQ on throughput.

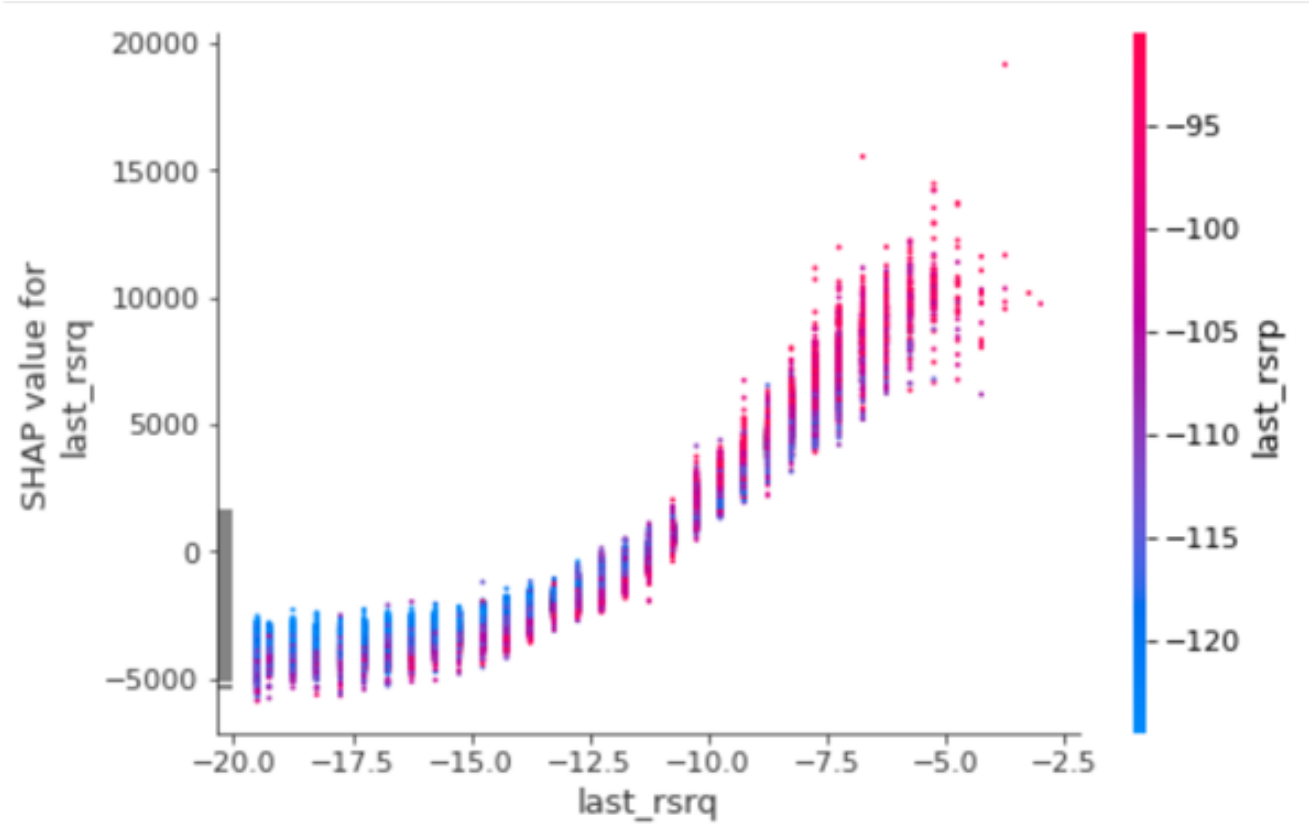

Figure 14: Reference Signal Received Quality (RSRQ) dependence plot

Figure 15 shows a linear relationship between Average Download Throughput given the specific value of SINR. This dependence plot shows a linear increase in expected contribution to throughput when moving from a low (i.e., -5) to high (i.e., 20) SINR. Again, this relationship is intuitive as a higher signal relative to the noise should indicate an improved downlink connection. When considering interactions with the metric Reference Signal Received Quality (RSRQ) which is displayed using a color gradient, the figure shows that higher values of RSRQ at the extremes of SINR will lead the model to rely more heavily on SINR to determine outcomes in throughput.

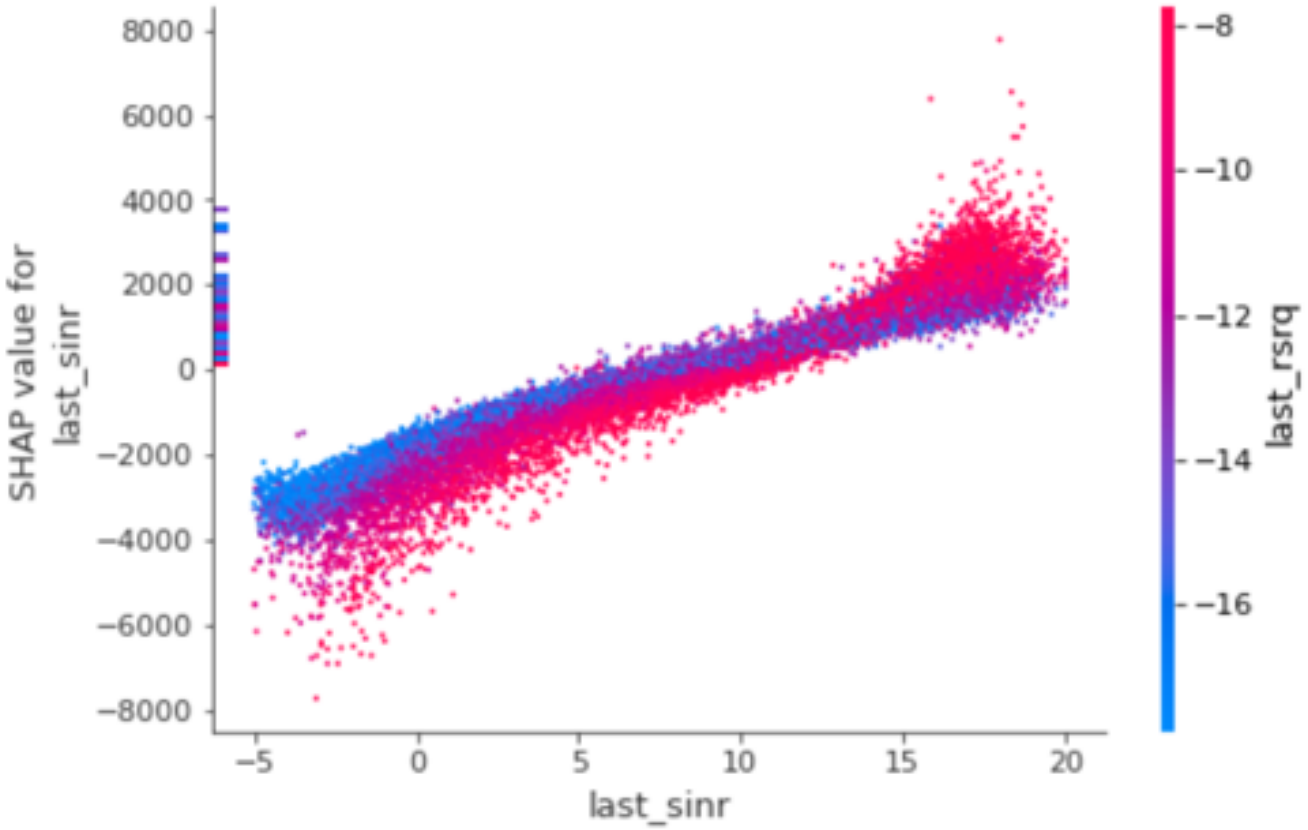

Figure 15: Signal-to-noise Ratio (SINR) dependence plot

While only two examples are shared here, many others were verified and confirmed to reasonable representations by the telecommunications experts assisting on this project. Overall, the results confirmed that the model could accurately represent and quantify known relationships between metrics in the system to throughput. By training a model over a larger area (e.g., CDRs of an operator for an entire city) we gain a representation of the expected performance for features across that area. While metrics like RSRP, RSRQ, SINR, TA will be available for any RAN network and telecommunications experts will have an understanding of how such metrics influence performance

of the network, learning directly from the CDRs in a specific network area allows the model to capture insights specific to the network of interest; some networks will be better or worse than others based solely on how modern their equipment is so having a model representation of the specific network will allow for improvement of problematic areas to improve performance. Put another way, what consists of a network problem may depend on the telecommunications operator and their budget.

Chapter 5

Conclusions and Future Work

This chapter includes conclusions drawn from the experiments and future work.

5.1 Conclusions

While a performance improvement by complex models such as gradient boosted tree over linear models was expected, the robust interpretability of these model thanks to the SHAP libraries was not. The creators of SHAP, most notably Scott Lundberg [10, 11, 9] created a theoretic robust interpretability strategy through by unifying Shapley Values with local model explanations. We incorporate this work into the fault analysis strategy proposed and tested in this Thesis.

5.1.1 Strategy

The strategy proposed involves first forming a representation of the data with a learning model. In this work, tabular data is represented using a tree-based model formed using gradient boosting. The learned model then encapsulates relationships and interactions between a set of inputs (e.g., CDR characteristics) and a target variable outcome (e.g., average throughput). Unlike linear correlations and expert rule based approaches traditionally used in certain industries like telecommunications, this model captures complex interactions by forming a non-linear mapping to connect inputs to target variable. Once the representation is formed, the proposed fault analysis strategy involves interpreting the model to gauge the relative importance of features or feature-value bins on negative outcomes in an isolated problem context.

The specific approach used to interpret the model is using SHapley Additive Explanations (SHAP) [10]. SHAP extends the computation of game-theoretic Shapley values to provide an

additive explanation of feature in a machine learning, fairly assigning feature contributions to model predictions.

Following the formation of a learned model, the fault analysis strategy uses SHAP to explain problematic samples. Thought of another way, an expert might ask a specific questions about the data such as "What explains low throughput in a particular region of the telecommunications network?" or "What explains dropped calls on a particular cell?". The analysis system then serves to answer the question when provided with a problematic sample that has been filtered for the appropriate constraints.

5.1.2 Assessment

Different experiments were conducted to evaluate the proposed fault analysis strategy. The experiments were performed on problematic samples drawn from a telecommunication network RAN 4G-LTE dataset. This dataset is composed of Call and Data Records (CDRs) transactions between users and the network, and a model is trained to learn how input features in the CDRs relate to average download throughput. SHAP was then used for model interpretation to provide relative feature importances to uncover how they relate to network degradation (i.e., low download average throughput).

From an implementation standpoint, gradient boosting with LightGBM was used for the dataset in this study, which was approximately 15M CDRs with high cardinality that included categorical variables. In particular, the standard GBDT implementation of LightGBM was used as it offered the best training times and representation, as measured by prediction error. The Tree SHAP implementation was used to improve training times, although empirical training times or differences in interpretation for alternate SHAP computation approaches were not yet explored.

The value of using a representative baseline as a control instead of a random baseline was validated for feature comparisons. This is relevant because training machine learning models requires a lot of computation, which can be impractical when performing analysis. In the experiment, problematic samples of CDRs were chosen on the basis of low average download throughput on the telecommunication network Cell. Two control samples were formed either by drawing from CDRs with the same cell Carrier EARFCN or a random sample. SHAP values were computed for all samples and Wasserstein w_1 distances between features in the problematic and control feature distribution were calculated. The results show large distances between SHAP distributions for most features when controlling for the Carrier EARFCN, which is explained by its limited bandwidth. Importantly for a fault analysis system, applying this control allowed for a new ranking of

feature importances; controlling for Carrier better prioritized feature relevance and abnormalities in the problematic sample.

Stability of the learning model and interpretation were also demonstrated. By forming 5 separate models in a typical k-fold validation setup, the stability of interpretation was tested against fixed samples of CDRs on problematic EnodeBs. Models were trained with different subsets containing $4/5^{th}$ of the training data and a different training order. The experiments demonstrated good stability of interpretation between model folds. Consistently, we observed only small variations of feature attributions between model fold. Categorical inputs with too few instances might warrant further investigation, however, as too few training examples having a specific characteristic will likely lead to instability for their representation. While a certain amount of variability was expected based on having different samples, the variability observed between model folds is minor compared to the clear differences between problematic samples. Thus, the fault analysis strategy is sufficiently stable to produce consistent results and detect differences between problematic samples.

Another experiment explored the implications of using the strategy for analytics instead of drawing statistical conclusions. The goal was to consider the implications for analysis when evaluating data the model has already seen vs unseen data. As this large, high-cardinality dataset proves difficult to overfit with a gradient boosted model, we explored the implications of using the learned model with interpretation to summarize the data it has seen before. In an experiment, we found that the median prediction error (i.e., the median over-prediction and under-prediction) are centered at 0 when evaluating seen data. This suggests that the model is fitting the data it has seen as much as possible given some constraint on the number of available parameters for the model. However, SHAP values distribute the relative contributions of features to the *prediction*, so having a systematic over-prediction or under-prediction may offer an incorrect uneven feature attribution. Thus, for the purposes of analytics where the goal is to evaluate the data and not the model, an argument can be made for applying the fault analysis to seen data.

Finally, in displaying the results we explored different visualization strategies. To facilitate analysis by domain experts, the flow-diagrams were used to quickly qualify differences in relative importances of features. This view can be expanded to showcase the impact of binned characteristics, quantiles for continuous metrics and categorical instances for dimensions of the data, so provide more precise visibility on the degradation observed in a sample. With one heuristic, we display what we call confidence bounds to provide visibility on the amount and direction of model error associated to predictions of a certain characteristics in the problematic sample. While there is not a good theoretical justification for *how* this approach was taken yet, the heuristic

provides a useful feedback when predictions associated with certain problematic features are being systematically misrepresented in a local sample. In general, review by telecommunications experts suggest that the strategy was effective as an analytics tool. Further, it was confirmed that known relationships were correctly captured by the model and explained using SHAP.

5.1.3 General Conclusions

There are many advantages to this strategy for fault analysis relative to conventional approaches. For instance, given a particular feature that occurs in a high proportion of problematic cases, it cannot be concluded that the feature is problematic as it may occur in an even higher proportion of normal cases. This is the fatal flaw of any strategy that only analyzes a small set of observations in a problematic sample while attempting to draw conclusions. With a large model trained on millions of normal and problematic cases, however, the model forms a more comprehensive representation of the dataset, determining how inputs relate to outcomes. The aim of this fault analysis strategy is, thus, to explain problematic samples by drawing from this general understanding of the dataset rather than local performance characteristics. This idea extends to the use of representative baselines with SHAP and distributions comparisons, which can be used to compare how problematic features are when compared to an appropriate baseline. Comparing samples and ranking the relative contributions of features provides a comprehensive way to perform fault analysis.

5.2 Limitation and Future Work

5.2.1 Application of Causality Approaches

An important next steps for this work would be to improve the ability to make causal claims as part of the analysis. Fault analysis in the telecommunication domain are commonly referred to as Root Cause Analysis, but this terminology was avoided as an acknowledgement that the model used in the strategy described in this Thesis can at most present features associations with problematic outcomes without providing any causal information. At this phase of the project, domain experts are relied upon to explore the results of the fault analysis system as they understand the interrelations of input features and are in a better position to infer possible causes based on correlational results.

The inclusion of causal graphs and modern causal analysis approaches will be essential to encapsulate expert knowledge for the relationships found between features in the dataset. This

use of causal graphs could ensure the discovery of reasonable outcomes and provide much better estimation of what is driving faults in problematic samples.

The application of counterfactual strategies as part of the system might also be a good next step to allow causal claims. In a traditional telecommunication setting, domain experts decide which configuration parameters to change when attempting to improve network performance. The decision is largely based on expertise and observations on the state of the network. In a complex system, however, a model representation of the data may be sufficient to predict the consequences of specific changes to the data if causal relationships are taken into account. Validating through counterfactual analysis and through guided interventions as experiments will be critical in extending the utility of the fault analysis system.

5.2.2 Testing with Predetermined Data Relationships

A machine learning model is formed by correlations and, consequently, relationships in the model may have been formed by so called "spurious connections" due to chance circumstances in the data. With enough data these spurious connections can dissipate, but there is no guarantee. Thus, in additional to using causality based approaches, using simulated data with predetermined and quantifiable relationships would aid in the process of detecting failure modes and revealing the ability of the fault analysis system to indicate and quantify problems for real-world data.

A recurring challenge in validation with telecommunications experts is that there is often no thorough or well-defined process when evaluating complex systems. Instead, data is often systematically filtered and reduced in dimension until only linear relationships are being investigated. Extending the experiments with simulated data will allow us to quantify whether, and how well, the fault analysis strategy will determine predetermined relationships in the data. The relationships are difficult to quantify, which is one of the primary motivations for the use of a machine learning model to learn relationships automatically.

5.2.3 Conditional Expectation

To further validate the approach, one next step will be to investigate whether differences between the conditional expectation used in TreeSAHP relative to the traditional marginal expectation version of SHAP poses a problem for the fault analysis system [19, 5]. The TreeSHAP algorithm is used for its practical implications in explaining many examples for a sample in a reasonable amount of time. Investigating these limitations will be helpful in bounding the possible use cases for fault analysis.

Chapter 6

Bibliography

[1] Nova passive agent. https://www.exfo.com/en/products/nova-passive-agent/, 2021. Accessed: 2021-06-02.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[3] Walter D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798, 1958.

[4] Plotly Technologies Inc. Collaborative data science, 2015. Accessed: 2021-07-02.

[5] Dominik Janzing, Lenon Minorics, and Patrick Bloebaum. Feature relevance quantification in explainable ai: A causal problem. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2907–2916. PMLR, 26–28 Aug 2020.

[6] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[7] Cassie Kozyrkov. Difference between analytics and statistics. https://www.kdnuggets.com/2019/09/difference-analytics-statistics.html, 2019. Accessed: 2021-04-18.

[8] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

[9] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):2522–5839, 2020.

[10] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[11] Scott M Lundberg, Bala Nair, Monica S Vavilala, Mayumi Horibe, Michael J Eisses, Trevor Adams, David E Liston, Daniel King-Wai Low, Shu-Fang Newman, Jerry Kim, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, 2(10):749, 2018.

[12] Yasuko Matsui and Tomomi Matsui. Np-completeness for calculating power indices of weighted majority games. *Theor. Comput. Sci.*, 263(1–2):306–310, July 2001.

[13] Christoph Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[14] K. Rashmi and Ran Gilad-Bachrach. Dart: Dropouts meet multiple additive regression trees. 05 2015.

[15] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "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*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[16] Ando Saabas. Dinterpreting random forests. http://blog.datadive.net/ interpreting-random-forests/, 2014. Accessed: 2021-05-21.

[17] Lloyd Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2.28:307–317, 1953.

[18] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11:1–18, 01 2010.

[19] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9269–9278. PMLR, 13–18 Jul 2020.

[20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.