A Tutorial on Data-Driven Quality of Experience Modeling with Explainable Artificial Intelligence

Nikolas Wehner*, Anika Seufert*, Tobias Hoßfeld*, Michael Seufert†
*University of Würzburg, Chair of Communication Networks, Würzburg, Germany
†University of Augsburg, Chair of Networked Systems and Communication Networks, Augsburg, Germany
{nikolas.wehner | anika.seufert | tobias.hossfeld}@uni-wuerzburg.de, michael.seufert@uni-a.de

***Abstract*—Data-driven Quality of Experience (QoE) modeling using Machine Learning (ML) is a key enabler for future communication networks as it allows accelerated and unbiased QoE modeling while autonomously adapting to changes over time. Traditional ML techniques only provide black-box QoE models, meaning that the models' internal reasoning is hard to understand. This makes it difficult not only to gain valuable insights into the data and the underlying QoE influence factors, but also to build trust in the predictions of the QoE model. To address these challenges and enhance performance, we identify four critical aspects for any data-driven QoE model: explainability, context adaptability, quantification of uncertainty, and data decentralization. In this work, we focus on eXplainable Artificial Intelligence (XAI), which is a promising solution to obtain explainable models allowing to address all critical aspects for data-driven QoE modeling. This tutorial introduces and explains relevant concepts from XAI and related fields. We apply these techniques to two realistic use cases, video streaming QoE modeling and web QoE modeling, for two different modeling tasks, classification and regression, and discuss their value in the context of XAI-based data-driven QoE modeling. In addition, to improve the reader's understanding, to ease the application of presented techniques for own use cases, and to foster research in the field of data-driven QoE modeling with XAI, we make Python Jupyter Notebooks available.**

***Index Terms*—Quality of Experience; Machine Learning; ML; Explainable Artificial Intelligence; XAI; Federated Learning; Uncertainty; Video Streaming QoE; Web QoE;**

I. Introduction

A. Motivation

While 5G mobile networks focused mainly on enhanced mobile broadband, massive machine type communication, and ultra reliable low latency communications, 6G is expected to advance mobile networks further by enabling human-centric [1] and Artificial Intelligence (AI) and Machine Learning (ML) empowered networking [2], [3]. As a consequence, Quality of Experience (QoE) is going to become a fundamental metric for quantifying the performance of 6G networks [1].

Understanding the factors influencing QoE [4] and subsequently building QoE models is therefore highly relevant for any service and network provider as it allows to avoid customer churn, and thus, revenue losses. To obtain such models, dedicated, subjective QoE studies have to be conducted in which participants rate the subjectively experienced quality of presented stimuli, i.e., representative content that would be experienced in realistic situations. This might include presenting videos with different number of stalling (rebuffering) events as stimuli, e.g., [5]–[13], or presenting web pages with different page load times as stimuli, e.g., [14]–[22], and obtaining the corresponding QoE scores in order to derive fundamental relationships between characteristics of the stimuli and the resulting QoE. However, such studies typically cover only some parts of the relevant parameter space due to time and budget constraints, and might be influenced by study design.

In [23], we therefore proposed data-driven QoE modeling, which comes with several benefits. These benefits are supposed to include a speedup of the modeling process, a reduction of the model bias introduced by study design and model selection, an increase in generalizability by merging heterogeneous data sources, and an automatic model refinement over time with newly available data. All that is required for such a modeling are large datasets composed of features (measurable characteristics of the experienced stimuli) and labels (QoE scores associated with these stimuli). Consequently, we face a classical supervised learning task that could be tackled with a rich toolbox of AI and ML techniques to obtain data-driven QoE models.

However, we must emphasize at this point that for any data-driven QoE model to be usable and valuable in practice, there is a crucial property: **Explainability**. QoE models deployed in practice should be explainable rather than black-boxes. Explainability provides valuable insights into data and QoE influence factors, while increasing the stakeholders' trust in the QoE model. Thus, this tutorial focuses on Explainable Artificial Intelligence (XAI) as promising solution for developing white-box models for data-driven QoE modeling.

We significantly extend our previous work [23] by providing comprehensive, accessible explanations for key XAI concepts. Moreover, we demonstrate XAI-based QoE modeling for two representative uses cases, video streaming and web QoE, explore the resulting explanations, and discuss their value for stakeholders. Going further beyond [23], we additionally identify, discuss, and analyze three beneficial aspects for the practical deployment and usage of data-driven QoE models.

Context adaptability: Data-driven QoE models should be able to incorporate context, such as user factors [4] or device differences [24], [25]. For instance, mobile users with smaller screens might perceive a lower video stream resolution less disturbing than TV users with the same video stream resolution. This tutorial demonstrates how a selected neural-network based XAI model can be adapted to account for contextual

factors by learning distinct functions for each context.

Quantification of uncertainty: Quantifying the uncertainty of a QoE model's predictions is crucial for stakeholders to avoid fatal business decisions, e.g., the black-box model misjudging QoE leads to poor QoE-aware network management and thus user churn. Hence, a data-driven QoE model should be able to express its uncertainty for a given prediction. Here, we integrate three uncertainty techniques into the same XAI model and subsequently discuss the quantified uncertainties and their implications.

Data decentralization: Centralized data collection requires large storage volumes, puts an additional burden on the network, and introduces privacy concerns by end-users. We highlight Federated Learning (FL) [26] as a solution, where the data remains on the local end-devices and model training is performed in a decentralized fashion. Therefore, future data-driven QoE models should support FL. We train the same XAI model in a FL setting, showing it can be trained decentralized while maintaining interpretability.

B. Fundamental Concepts

Next, we introduce and discuss fundamental concepts required for the remainder of the work. First, we formally define QoE and afterwards show how QoE can be inferred from the network and/or the application. Ultimately, we introduce XAI with respect to explainability and interpretability and discuss a general taxonomy of XAI methods.

1) Quality of Experience: For a long time, network performance and resulting user satisfaction have been quantified with the concept of Quality of Service (QoS) [27]. This concept defines the performance level of a service using objective, measurable technical parameters such as bandwidth, latency, jitter, and packet loss. QoS thus describes the efficiency and reliability of a network or system to deliver data (for a service) [27]. However, QoS does not directly reflect true user perception, resulting in challenges when mapping objective parameters to subjective user experiences.

Thus, QoE was introduced to describe the subjectively perceived quality of a service [4], complementing QoS. More formally, QoE is defined as the "*the degree of delight or annoyance of the user of an application or service*" [4]. Hereby, user's QoE may be affected by a plethora of influence factors, which are defined as "*[a]ny characteristic of a user, system, service, application, or context whose actual state or setting may have influence on the Quality of Experience for the user*" [28]. These influence factors include human factors, e.g., demography or the user's emotional state, system factors, e.g., content and network quality, and context factors, e.g., physical and temporal contexts. Note that the applicability of specific influence factors strongly depends on the considered service.

A plethora of metrics for quantifying QoE have been proposed in literature, with Hoßfeld et al. [29] providing a taxonomy for classifying metrics based on subjective measurements. These metrics can be generally categorized into opinion-based and behavioral-based metrics. Opinion-based metrics are directly obtained from users and can be further classified into metrics related to overall quality, perceptual quality dimensions, difficulty in usage, acceptance, or early termination. In this work, we focus on metrics related to overall experience consisting of user opinions/ratings on the predefined Absolute Category Rating (ACR). With the ACR scale, participants rate the quality of their experience with a stimulus on a scale from 1 (bad) to 5 (excellent) [30]. Afterwards, QoE is quantified by computing the Mean Opinion Score (MOS) of these ratings. The MOS is the most commonly used metric to express QoE, and is computed by averaging individual user ratings, e.g., on the ACR scale, for a given stimulus, thus representing the average user-perceived performance.

2) QoE Modeling & Monitoring: Figure 1 provides a high-level overview of QoE modeling and monitoring, highlighting the stakeholders involved in the QoE delivery chain. A server delivers a service and content to a user via the Internet and an access network, such as a cellular network, with typically encrypted network connections. These services are accessed through the user's application running on an end-device, such as a smartphone. The user's perception of the application performance defines the user's QoE. In subjective studies, users provide ratings to quantify their experience, providing direct feedback to the performance of the service provider, network operator, or both. Since service providers and network operators want to optimize and monitor performance with respect to QoE, they require methods to estimate QoE during deployment without relying on user feedback. This is achieved by developing objective QoE models that estimate QoE based on measurable application-layer and/or network-layer parameters, and optionally context factors, e.g., device type or screen resolution.

Application-layer parameters typically resemble known QoE influence factors from research and may include video Key Quality Indicators (KQIs) like initial delay, stalling, or video quality, or web browsing KQIs like loading times, page sizes, or number of Hypertext Markup Language (HTML) objects. Due to encryption, these parameters are only measurable by the service provider, but not by the network operator. Consequently, the network operator is limited to measuring network Key Performance Indicators (KPIs), such as throughput, volume, or latency, which makes direct QoE estimation a more challenging task compared to using application-layer parameters [31], [32]. Thus, indirect QoE modeling could be required, where QoE influence factors are first approximated from network-layer parameters and then fed into the QoE model [33].

Designing such a model can be framed as a supervised task, where the objective is to map features (measurable application-layer or network-layer parameters) to a QoE metric, such as the MOS. If the labels are continuous, the task is a regression problem. If the labels are categorical (multiple classes), it becomes a classification task. First, the monitored parameters must be converted into a set of features that accurately describe the observed service and network performance. In QoE modeling, these features are typically organized as tabular

This article has been accepted for publication in IEEE Communications Surveys & Tutorials. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/COMST.2025.3583227

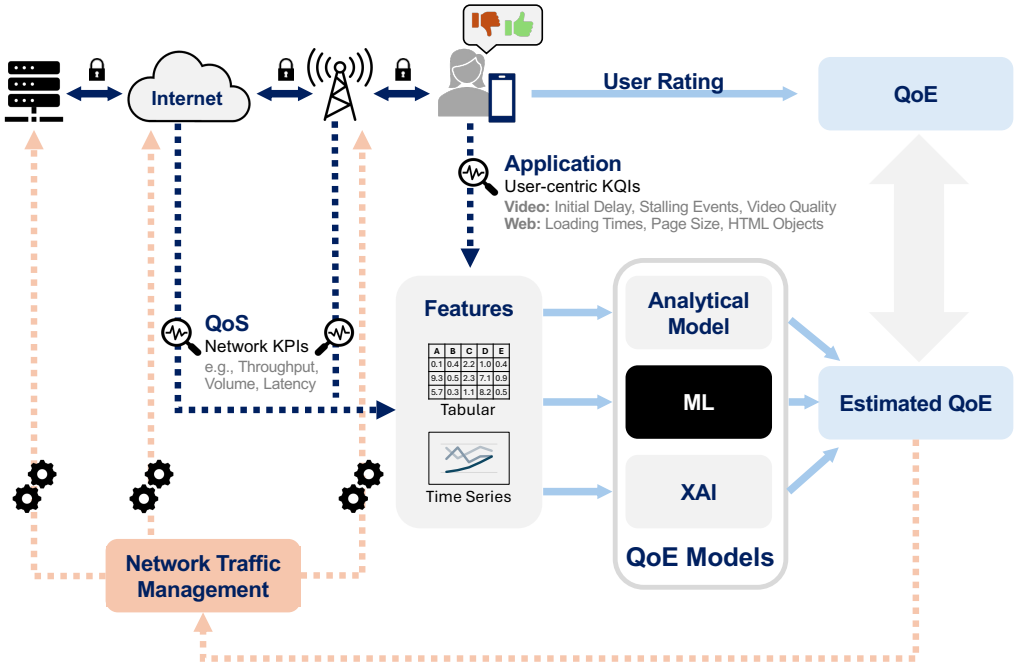


Fig. 1: Big picture of QoE modeling, monitoring, and QoE-aware network traffic management.

data, where each row represents a single observation with its corresponding QoE ground truth (label). Alternatively, features can be presented as a time-series, where a single observation is split into multiple time steps, e.g., 1 second intervals, each with its own QoE ground truth or the ground truth for the entire time-series. This work focuses on tabular data, as it is easier to implement for deployment and avoids the additional explainability challenges of time-series data. The approaches discussed in this tutorial apply whenever input data can be transformed to tabular data, allowing for transferability across various QoE or other use cases. The model is first developed during a training phase, where QoE ground truth is available. In this phase, the model learns to map the features to QoE metrics. Once trained, the model can be deployed to estimate QoE for samples measured in real-world conditions, where ground truth may not be available.

Traditionally, QoE modeling used either analytical models, for example, based on curve fitting, or black-box ML models to estimate QoE. While analytical models are interpretable, they often introduce biases from the research team and are more challenging to develop. In contrast, data-driven ML models are typically black-boxes, obscuring their reasoning and reducing stakeholder trust. To address this, we propose XAI as a solution, enabling data-driven QoE modeling with explanations for its reasoning.

Further, we want to remark that this work does not focus on real-time QoE monitoring. These approaches require dedicated hardware and pipelines, such as Intel Tofino P4 Switches, for feature measurement and low model inference times [34]–[37]. However, such specialized high performance pipelines are beyond the scope of this work. Instead, the presented approaches do not explicitly focus on the timely delivery of monitoring information, but are suited for all kinds of service and network management.

3) QoE-Aware Network Traffic Management: Traditionally, network operators utilized network management to meet Service Level Agreements (SLAs) and optimize resource usage, focusing primarily on the efficient transmission of packets and flows. Today, QoE-aware network traffic management [38] shifts the focus towards enhancing end-user satisfaction by utilizing cross-layer information, e.g., application-layer metrics, and collaborative mechanisms, e.g., client-network communication. Its primary goal is to optimize network resource utilization while improving overall QoE and ensuring QoE fairness across users and applications.

Such network management mechanisms require continuous monitoring of network conditions and the collection of both KPIs and KQIs. These metrics are compared against targets derived from SLAs and user expectations. While SLA compliance is simpler to verify, assessing user satisfaction is more complex, as it can not be directly measured and must instead be inferred from KPIs and KQIs. However, KPIs and KQIs alone are typically insufficient to infer QoE, but usually represent only parts of the user experience. Accurate QoE estimations thus depend on accurate and trustworthy QoE models, the focus of this work.

If SLA or QoE expectations are not met, the network operator initiates appropriate network traffic management actions,

e.g., routing optimizations, traffic prioritization, bandwidth shaping, or signaling applications to adapt their demands. These actions are enforced through configuration updates within the different nodes of the network infrastructure. As illustrated in red in Figure 1, the QoE estimated by the QoE model can directly inform such management decisions. In principle, the management can be applied throughout the delivery chain, e.g., from servers or CDNs to the general Internet and access networks.

Ultimately, the goal of QoE-aware network traffic management is the creation of a dynamic control loop that aligns network performance with QoE, thus bridging the gap between technical efficiency and perceived service quality.

4) Explainable Artificial Intelligence: XAI is a research field focusing on making AI systems understandable to human users [39]. Understanding black-box AI systems is fundamental for all parties as such models are supposed to align with ethics, avoid biases, and generally increase trust [40]. Subsequently, XAI aims to develop methods able to provide explanations on black-box model behavior or to develop interpretable models directly. Although explainability and interpretability are often used interchangeable in the literature, it is important to distinguish between these concepts [41].

Explainability focuses on providing explanations for a black-box model to help users understand why the model produced a specific output. Formally, an explanation is additional information made available by the model itself or by an external algorithm, e.g., the importance or relevance of inputs for the model's predictions [40]. These explanations aim to approximate the model's internal processes and are supposed to provide users insights into how the model works. These explanations often include feature importance, heatmaps indicating the features a model relied on for its prediction, or rules approximating its predictive behavior [40].

In contrast, interpretability allows the user to study and clearly understand the mathematical processes within the model transforming the inputs to the respective outputs, thus providing transparency by clarifying how the algorithm actually works [40], [42]. As a result, interpretable models often provide mathematical formulations that efficiently describe the learned behavior of the model.

These differences are also evident in literature where taxonomies of XAI methods are discussed [39]–[41], [43]. Broadly, XAI methods are classified into *interpretable* and *post-hoc* explanation techniques. Interpretable models, also called in-hoc, transparent, white-box, or intrinsic models, are interpretable by design to make their learned behavior and reasoning understandable. In contrast, post-hoc explanation techniques are applied after training a black-box model and aim to explain the black-box behavior. As we will see, many interpretable models and post-hoc techniques rely on additive functions, which are easily understandable by humans.

Post-hoc explanation techniques can be further categorized in *model-agnostic* and *model-specific* techniques. Model-agnostic techniques can be applied to any kind of model, while model-specific post-hoc explanation techniques are restricted to a certain kind of model. An example for a model-specific post-hoc explanation is the saliency map of Convolutional Neural Networks (CNNs), which shows where the model focused the image to generate its prediction, but which can not be computed for Random Forest (RF), an ensemble of trees, for example. Post-hoc explanation techniques are usually either based on perturbations, i.e., randomly changing the input space to observe what happens with the output, or on gradients, i.e., the influence of individual features can be estimated when using differentiable functions like neural networks.

Finally, the scope on which XAI techniques and models operate can be used to distinguish between them. There are models which provide *local, global, or both kinds of explanations*. Local explanations are provided for a single instance only and usually differ for each instance. They provide us with a local view of the model's knowledge. Global explanations on the other hand provide a general idea on what the model learned independently of the instances. For more details on the general taxonomy we refer the interested reader to the works of Arrieta et al. [43] and Molnar [44].

In this tutorial, we investigate the applicability of both post-hoc explanation techniques and interpretable models to QoE modeling.

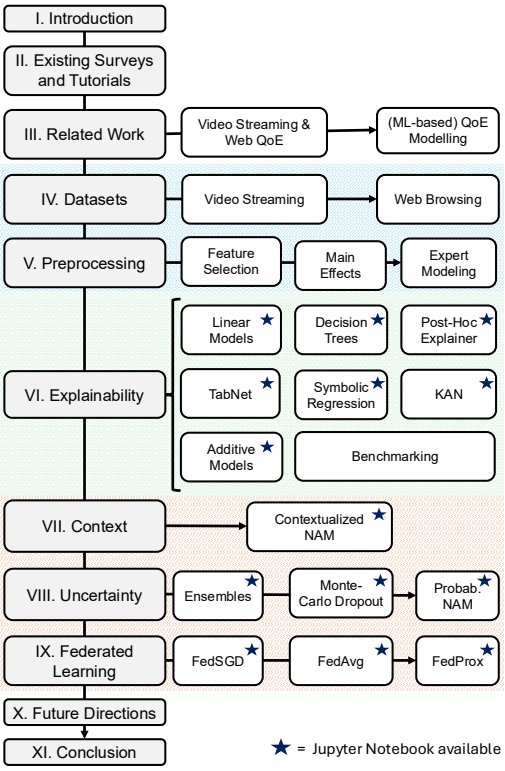


Fig. 2: Structure of this work.

C. Structure

Figure 2 shows the structure of this work: Existing surveys and tutorials and other related work are discussed in Sections

II and III. Additionally, we provide acronyms and abbreviations relevant for the remainder of this work in Table I to support readability. The datasets used for this tutorial as well as a short discussion on the preprocessing, the main effects of the datasets, and expected relationships based on expert modeling are presented in Sections IV and V. We start Section VI by introducing general common ML concepts, before we dive into different, existing XAI techniques and models. We begin with simple XAI models and continuously increase the complexity. For each model we first introduce the basic model concepts to the reader, then apply the model to our datasets and discuss the obtained explanations, before we talk about the learned lessons. In the following sections, we focus on a single selected model only, the Neural Additive Model (NAM). We showcase how this model can be extended to incorporate context (Section VII) and uncertainty (Section VIII). Here, we also first introduce relevant concepts to the reader for both context and uncertainty, before applying different techniques and investigating the obtained explanations. In Section IX we then explain federated learning and exemplary aggregation strategies, before we investigate the suitability of NAM for federated XAI-based QoE modeling. We discuss potential future directions of XAI-based QoE modeling in Section X, before this tutorial is concluded in Section XI.

All models presented in this tutorial are implemented in Python. We make Jupyter Notebooks publicly available[1], which will not only help to better understand the explained concepts and their usage, but which will also enable researchers to perform rapid prototyping on their own QoE datasets. They consist of standalone, executable scripts that include extensive documentation, each focusing on a specific technique covered in Sections VI, VII, VIII, and IX, as highlighted in Fig. 2. While we target QoE modeling in this work, we also want to emphasize that all presented concepts and techniques can also be transferred to other network management use cases, e.g., network security and intrusion detection, monitoring of Internet of Things (IoT) devices, or network capacity planning.

TABLE I: List of Abbreviations.

Abbreviation	Description
ACR	Absolute Category Rating
AI	Artificial Intelligence
ATF	Above-The-Fold
BI	Byte Index
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
CSS	Cascading Style Sheets
DASH	Dynamic Adaptive Streaming over HTTP
DT	Decision Tree
EBM	Explainable Boosting Machine
FedAvg	Federated Averaging
FedProx	Federated Optimization in Heterogeneous Networks
FedSGD	Federated Stochastic Gradient Descent
FL	Federated Learning
GAM	Generalized Additive Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IQX	Exponential Interdependency of QoE and QoS
KAN	Kolmogorov-Arnold Network
KPI	Key Performance Indicator
KQI	Key Quality Indicator
LIME	Local Interpretable Model-Agnostic Explanations
LinReg	Linear Regression
LLM	Large Language Model
LogReg	Logistic Regression
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multilayer Perceptron
MOS	Mean Opinion Score
MSE	Mean Squared Error
NAM	Neural Additive Model
OI	Object Index
PLT	Page Load Time
QoE	Quality of Experience
QoS	Quality of Service
RF	Random Forest
RMSE	Root Mean Square Error
SHAP	SHapley Additive exPlanations
SI	Speed Index
SLA	Service Level Agreement
SOS	Standard Deviation of Opinion Scores
SR	Symbolic Regression
SROCC	Spearman's Rank Correlation Coefficient
TTFB	Time to First Byte
TTLB	Time to Last Byte
VoIP	Voice over IP
VR	Virtual Reality
WQL	Logarithmic Relationship of Waiting time and QoE
XAI	Explainable Artificial Intelligence
XGBoost	eXtreme Gradient Boosting

II. Existing Surveys and Tutorials

On the topic of QoE as well as on XAI a plethora of surveys and tutorials already exists. However, there is no work that combines both topics as presented in this work. Existing studies dealing with QoE and XAI are not as comprehensive as this paper. We provide an overview on available literature in Table II. The research works are categorized according to their focus on QoE, XAI, and the considered services, along with a summarized description of the core contents and the publication year.

The closest surveys to our work are the works from Barman et al. [13] and Kougioumtzidis et al. [45]. Barman et al. [13] survey existing works on QoE modeling in the context of Hypertext Transfer Protocol (HTTP) adaptive video streaming. While they also discuss works that use ML to model different Dynamic Adaptive Streaming over HTTP (DASH) influence factors, none focus on XAI and its application to QoE modeling. Moreover, our work differs in that we additionally consider web QoE use cases, and in that we present methodologies that are transferable to any QoE use case. Kougioumtzidis et al. [45] survey existing approaches for assessing QoE with ML but do not focus on XAI, too. Most of the discussed works rely on black-box models and only a few works utilize simple interpretable models like linear regression or Decision Trees (DTs). While these basic models can technically be considered XAI, they are typically used for exploratory data analysis and are not comparable to the more sophisticated XAI methods covered in our tutorial. The closest

[1]https://github.com/lsinfo3/comst-xai-qoe

TABLE II: Overview on existing surveys and tutorials in the field of QoE and XAI, where ✓ corresponds to match, ✗ corresponds to no match, and (✓) corresponds to a partial match.

Type	QoE	XAI	Services/Data	Summary	Year	Work
Survey	✓	✗	Multimedia Services	ML-based QoE Assessment	2022	[45]
	✓	✗	Video Conferencing	QoE of Telemeetings and Video Conferencing	2022	[46]
	✓	✗	Unified Communications	QoE for Unified Communications (Audio, Video, Text)	2020	[47]
	✓	✗	Video Streaming	QoE Modeling	2019	[13]
	✓	✗	Multimedia Services	Trends in QoE Management	2018	[48]
	✓	✗	Web Services	Web QoE	2017	[18]
	✓	✗	Video Streaming	Measuring QoE of Video-on-Demand Services	2015	[11]
	✓	✗	Video Streaming	HTTP Adaptive Streaming	2014	[9]
	✓	✗	Cloud Services	QoE in Cloud Services	2014	[49]
	✓	✗	Generic	ML-based QoS/QoE Correlation Models	2014	[50]
	✓	✗	Multimedia Services	QoS/QoE Correlation Models	2013	[51]
	✓	✗	Voice over IP (VoIP)	QoE of VoIP	2012	[52]
	✗	✓	Networking	Traffic Classification and Prediction & Intrusion Detection	2024	[53]
	✗	✓	Networking	XAI for 6G	2024	[54]
	✗	✓	Networking	XAI for 6G O-RAN	2024	[55]
	✗	✓	Networking	XAI for 5G	2024	[56]
	✗	✓	Networking	XAI for IoT Security	2023	[57]
	✗	✓	Networking	XAI for IoT	2023	[58]
	✗	✓	Networking	XAI for Cybersecurity	2023	[59]
	✗	✓	Generic	XAI and Natural Language Explanations	2023	[60]
	✗	✓	Generic	Taxonomy, Challenges & Future Research	2022	[61]
	✗	✓	Time-Series	XAI for Time-Series Data	2021	[62]
	✗	✓	Tabular	XAI for Tabular Data	2021	[63]
	✗	✓	Generic	Taxonomy & Responsible AI	2020	[43]
	✗	✓	Generic	Taxonomy & Software	2020	[40]
	✗	✓	Generic	Taxonomy	2020	[64]
	✗	✓	Generic	Taxonomy	2018	[39]
	✗	✓	Generic	Taxonomy	2018	[65]
Tutorial	✓	✗	Video Streaming	ML Pipeline & Deployment in 5G/6G Networks	2021	[66]
	✓	✗	Multimedia Services	QoE Management	2019	[67]
	(✓)	✗	Video Streaming	From QoS to QoE for Video	2014	[8]
	✗	✓	Generic	Mostly Post-Hoc Explanation Techniques	2021	[68]
	✗	(✓)	Generic	Fuzzy Modeling	2018	[69]
Tutorial	✓	✓	**Video Streaming & Web Browsing**	**XAI-based QoE Modeling only**	**2024**	**Ours**

tutorial to ours is the work from Chen et al. [8]. The authors present a tutorial on video quality assessment and discuss how to obtain QoE from QoS in the context of video streaming. Their work is close to ours in that they also discuss data-driven QoE modeling in the end and also propose to use simple interpretable models such as linear models and DTs for this purpose. However, a variety of novel XAI techniques have been proposed since their 2014 publication. These new XAI techniques require validation for the QoE use case, a gap we aim to close in this work.

The other surveys concerning QoE mostly focus on specific services, e.g., video conferencing [46], cloud services [49], web services [18], or video streaming [9], and entirely neglect ML and XAI. In contrast, the XAI surveys mostly discuss taxonomies on how the different models and techniques can be categorized and which approaches exist. In addition, some surveys focus on specific kinds of data, e.g., time series data [62] or tabular data [63]. Further, a plethora of surveys discussing XAI and networking exists. These surveys focus on different aspects of networking, such as traffic classification [53], IoT [58], security [57], [59], and 5G/6G in general [54]–[56]. Our tutorial complements these works by exploring the capabilities of XAI to model the user-centric perspective of networking through QoE.

The tutorial by Ahmad et al. [66] is also relevant to our work. The authors present a tutorial on how to create a ML pipeline for QoE prediction of video streaming and how to deploy such ML models in 5G or 6G networks with the help of Software Defined Network, Network Function Virtualization, and Multi-Access Edge Computing. It allows that we omit these preparation steps, or touch them only lightly, respectively, and instead we refer to the survey of Ahmad et al. for more information.

Based on our literature review, we thus conclude that, despite its immense potential, data-driven QoE modeling with XAI has not been extensively covered and used in research yet. We therefore aim to close this gap with this tutorial. As we focus on the use case of video streaming QoE and web QoE as well as on QoE modeling in the contexts of ML, federated learning, and uncertainty in this work, we also discuss relevant related work on these topics in the following.

III. Related Work

A. Video Streaming QoE

Several QoE models have been proposed for Internet services like video streaming [9], [13]. As shown in [13], there

are various video QoE models, which consider different inputs, e.g., media-layer inputs like decoded audio/video signal, parametric inputs like packet headers, bitstream information like quantization parameter, as well as hybrid models, which consider a combination of the aforementioned inputs. In this work, we compare the performance of our data-driven video streaming QoE models to the expert QoE models ITU-T P.1203 [24], Hoßfeld [70], Liu [71], Mao [72], Mok [73], Petrangeli [74], and VsQM [75]. The majority of these models considers the total stalling length, the number of stalling events, and the visual quality (bitrate and/or resolution) as important QoE influence factors. Note however that Hoßfeld and Mok do not consider adaptive video streaming. P.1203, Mao, Mok, and VsQM also factor in the initial delay. P.1203 additionally considers the number of video quality switches and it is the only expert model, which also incorporates ML. In [76], the authors showed that many of those video streaming QoE models perform significantly different as they attach different weights to the QoE influence factors. This suggests that a data-driven approach to QoE modeling can bring benefits to the QoE community in terms of model accuracy and generalizability.

Video streaming places substantial demands on network resources, while network performance directly affects KQIs of the viewing experience. Adaptive video streaming follows a characteristic on-off pattern, with short traffic bursts for segment downloads followed by idle periods [77], [78], posing challenges for both applications and network. The initial delay is mainly influenced by network latency and available bandwidth during playback startup. Reducing this delay requires network operators to provision sufficient resources at startup, directly impacting QoE. However, this creates traffic spikes that must be carefully managed to prevent network congestion. Stalling is strongly linked to insufficient or unstable throughput, often caused by bandwidth fluctuations, high latencies, and packet loss, resulting in buffer depletion. Network operators should thus ensure stable throughput and low latency to minimize the risk of stalling. Visual quality depends on the available throughput, where higher throughputs enable the delivery of segments with better quality. Conversely, bandwidth constraints force lower video quality to avoid stalling. Subsequently, delivering high video quality increases network load. Finally, quality switches reflect network instability. As throughput fluctuates, adaptive bitrate algorithms adjust video quality to balance resolution and stalling risk. Frequent switching may degrade QoE, making it essential for network operators to provide a stable throughput when aiming to support excellent streaming. Since all these factors affect both video streaming QoE and network performance, accurate and explainable video streaming QoE models are essential for network operators and service providers to ensure high video streaming QoE and gain insights for efficient QoE-aware network traffic management via control loops.

B. Web QoE

In the context of web QoE, loading times are considered fundamental for estimating web QoE. Several web QoE models have been proposed [18]. To capture these loading times and thus QoE, several metrics have been suggested in literature. These metrics are usually classified as either time-instant metric or time-integral metric. Time-instant metrics denote a specific event, at which a user may perceive a web page as fully loaded. The most prominent time-instant metric is the Page Load Time (PLT), which corresponds to the time, when a web page is fully downloaded. Some other time-instant metrics are Time to First Byte (TTFB) and Time to Last Byte (TTLB), i.e., the points in time, when the first or last byte of a web page are received, respectively. Time-integral metrics include Google's Speed Index (SI) and its variations, e.g., Byte Index (BI) and Object Index (OI) [79]. These metrics compute an integral over the complementary progress in the viewport based on different attributes, e.g., visual progress for SI, byte progress for BI, and object progress for OI. To derive a web QoE model from these metrics, the metrics are usually aligned to the Exponential Interdependency of QoE and QoS (IQX) [80] or the Logarithmic Relationship of Waiting time and QoE (WQL) [15] hypotheses [19], [21], [22]. The IQX hypothesis assumes an exponential relationship between metric and web QoE, while the WQL hypothesis assumes a logarithmic relationship. However, the authors of [81] reveal that the distribution of the user-perceived PLT is often multimodal, so IQX and WQL hypotheses are likely not sufficient to fully capture QoE. Instead, a multidimensional approach to QoE modeling is required, e.g., in the form of multidimensional IQX [82], an extension to the IQX hypothesis. However, here our data-driven approach might also assist by generating a context-aware QoE model.

Loading times like PLT and SI fundamentally depend on network latency, available bandwidth, and packet loss. High round-trip times or limited throughput can significantly delay the retrieval of web objects and thus the rendering of web pages. To mitigate these effects, network operators and service providers should aim to minimize latency and optimize bandwidth usage. This is especially important for TTFB, which reflects the perceived responsiveness of the server and is characterized by DNS resolution, TCP/TLS handshake, and the initial data transfer. While web browsing consumes fewer network resources than video streaming, it remains the most widely used application and often serves as a gateway to other services. Nevertheless, user behavior can introduce unpredictable traffic patterns, making it essential to understand web QoE and proactively adapt QoE-aware network management via control loops accordingly. To this end, accurate and transparent web QoE models are essential, offering network operators and service providers guidance on how to orchestrate their networks while ensuring high QoE.

C. (Federated) ML-based QoE Modeling

The survey in [45] shows that ML-based QoE modeling in multimedia systems is already widely used, including Virtual

Reality (VR), 360 degree video, and gaming. However, the QoE models are based on shallow learning methods, e.g., RFs, or on deep learning methods, which lack explainability. Thus, it is difficult to understand what QoE factors are relevant and how they affect the QoE score. Most similar to our work is the work of Fiedler et al. [83] where model trees are used to estimate points of change in video streaming QoE. For this specific class of model trees, the leafs correspond to linear functions and are thus explainable.

There also exists research on collaborative (or federated) QoE modeling. Porcu et al. [84], [85] use clustering and federated learning to apply QoE modeling to a public media content dataset [84] and to a public web QoE dataset [85]. Ickin et al. [86] also model video streaming QoE with collaborative learning using the Waterloo III video streaming QoE dataset [87], one of the datasets also used in this work.

D. Uncertainty in QoE Modeling

Uncertainty in QoE modeling has also been investigated in other works. In [29], the authors propose different means to quantify uncertainty from the score distributions. The proposed measures include the standard deviation and subsequently the Standard Deviation of Opinion Scores (SOS) hypothesis [88], entropy, the Cumulative Distribution Function (CDF), and quantiles. The authors of [89] also quantify uncertainty for assessing the QoE of Voice over IP (VoIP) calls. To achieve this, they learn a Bayesian Network, which considers different context factors. Another field of research, which requires the estimation of uncertainty, is active learning [82]. With active learning, the next data point added to the training dataset has to be determined. One strategy to decide on the next sample is to exploit uncertainty. The authors of [82], therefore, select the next sample with the goal to minimize overall prediction variance and, thus, decrease uncertainty (similar to [90]). In contrast to previous works, data-driven QoE modeling allows to implicitly learn uncertainty along with the actual task.

E. ML and XAI-based QoE Monitoring

While related work for explainable ML-based QoE modeling is scarce, many works have addressed (explainable) ML-based QoE monitoring. In these works, the goal is to directly infer QoE, e.g., P.1203 [24] or QoE influence factors, e.g., stalling or video quality from QoS parameters. Our work thus addresses the fundamental research, which makes these works possible. The authors of [91]–[93] for example use mostly black-box ML models to estimate video streaming QoE influence factors like stalling and video quality from QoS. For the web, the authors of [16], [35], [94] also use various ML models to estimate web metrics like PLT and, subsequently, web QoE.

In contrast to previous works, the authors of [95] present an explainable classifier based on logistic regression to improve stalling prediction. Schwarzmann et al. [36] use regression techniques to estimate QoE for the use cases of video streaming and VoIP in 5G networks. They rely on linear regression techniques like Ridge Regression and Lasso (Least Absolute Shrinkage and Selection Operator) Regression, which are explainable by default. The authors of [96] use fuzzy models to derive rules which explain why the model mapped QoS parameters to one of three classes representing stalling.

We thus observe that XAI has been already used for QoE monitoring, even though scarcely. We want to emphasize that while we mainly focus on QoE modeling in this work, all presented XAI techniques can be also transferred to the task of understandably inferring QoE from QoS, i.e., modeling the relationship between the network and the user experience in an understandable fashion. As a consequence, our work can be also used as a guideline for XAI-based QoE monitoring.

IV. Datasets

A. Use Case: Video Streaming

We crawled various publicly available sources for video QoE databases meeting specific criteria. In particular, we looked for databases containing rich information about experiment conditions and stimuli, which could have an impact on video streaming QoE, and thus, can be used as features for the ML models. This includes, for example, various video streaming KPIs, such as stalling information and visual quality. Moreover, the database was required to provide the corresponding subjective user ratings for all stimuli, which we require as labels to train ML models in a supervised fashion.

Five databases [24], [87], [97]–[99] matched our criteria, and we preprocessed and aggregated them into a new dataset consisting of 2,571 video streaming sessions. We characterize the databases shortly in Table III and observe that our new, merged dataset is highly heterogeneous due to the different databases. This can be seen, for example, in the duration of the video clips ranging from 10 up to 240 seconds. We can also see that the amount of instances containing an initial delay and stalling events vary between the databases. In addition, the MOS for the Waterloo databases [87], [97] is significantly higher compared to the other three databases.

For each video of a database, we extracted the initial delay in seconds, the number of stalling events, the total stalling duration in seconds (without initial delay), the playback duration in seconds (excluding stalling and initial delay), and the video width and height in pixels, and used them as features to describe the stimulus. Additionally, we extracted the bitrate (Mbps), the frame rate, and the stalling duration for every second of the video duration. Considering the distributions of these per-second KPIs in a single session, we computed the mean, standard deviation, minimum, and maximum statistics, which we also use as features. Finally, we also extract the number of quality switches, and the recency time of the last quality switch, but not the quality levels themselves since they are not compatible across the databases due to different minimum and maximum resolutions. Constant features, e.g., the frame rate statistics, were immediately removed from the feature set. Finally, we consider the continuous MOS of each stimulus as label, which shall be predicted by the QoE models. Note that we normalize all MOS scores across all databases to an ACR scale from 1 to 5.

TABLE III: Characteristics of considered video QoE databases.

Database	# Sessions	# Videos	Duration [s]	Initial Delay Incl.	Stalling Incl.	Avg. Stalling Length [s]	# Avg. Stallings	Avg. Bitrate [Mbps]	MOS (Std.)
Waterloo III [87]	450	20	10	100%	49.1%	3.2	2.4	2.52	3.43 (0.62)
Waterloo IV [97]	1350	5	28	12.3%	23.9%	3.6	2.0	4.60	3.65 (0.64)
LIVE Netflix I [98]	112	14	60-87	62.5%	50.0%	8.4	1.3	0.24	3.00 (0.70)
LIVE Netflix II [99]	420	15	25	0%	39.5%	2.7	2.5	0.60	2.99 (0.82)
ITUT-T Rec. P.1203 [24]	239	157	60-240	30.1%	43.9%	12.9	1.4	1.77	3.15 (0.97)
Merged dataset	**2571**	**211**	**10-240**	**29.48%**	**33.87%**	**4.8**	**2.1**	**3.13**	**3.43 (0.76)**

Overall, this use case serves as an example of mapping application parameters (user-centric KQIs) directly to QoE (cf. Fig. 1). Even though networking aspects are not directly involved in the QoE modeling process of this scenario, accurate QoE modeling of application parameters can still reduce the burden on the network. This can be achieved by implementing network management mechanisms that optimize and prioritize traffic more effectively. Such optimizations can lead to smarter bandwidth allocation and lower latencies for high-priority applications, guaranteeing a smoother user experience. Further, general application optimization typically yields an improved network performance by enabling a more efficient use of network resources, thereby reducing congestion, and promoting fair bandwidth distribution among competing users.

B. Use Case: Web Browsing

To showcase the generality and transferability of the approach, we also evaluate the performance of XAI-based modeling for a web QoE use case. In contrast to video streaming QoE, the availability of suitable web QoE datasets is much more limited. We identified only two suitable datasets, namely, the dataset from Schatz et al. [100] and the dataset from da Hora et al. [101].

The dataset from Schatz et al. [100] contains demographic information as well as subjective ratings for different web browsing scenarios, where different websites are accessed in different network scenarios. For the network scenarios, uplink and downlink bandwidth and round trip times are manipulated. The dataset contains a total of 572 subjective ratings.

In contrast, the dataset from da Hora et al. [101] contains 3,010 subjective ratings for a highly diverse set of websites, but no demographic information. In total, 241 participants rated their browsing experience on ACR scale in an experiment setup, where participants had to use Google Chrome to browse to 12 popular non-landing web pages present in the Alexa Top 100 of France. As with the other dataset, the experiment setup allowed to manipulate the network conditions with respect to latency and packet loss. Additionally, the dataset provides a rich set of application layer features, e.g., the amount of bytes in Cascading Style Sheets (CSS), JavaScript, image, and video objects contained in the website, or the number of contacted domains during the web page request as well as several relevant web QoE metrics, e.g., PLT, Above-The-Fold (ATF), BI, OI, TTFB, and TTLB.

As both datasets use completely different features, we can not merge them into a new heterogeneous dataset. Therefore, we only select the dataset from da Hora et al. [101] for our showcase as it contains more subjective ratings and provides a more diverse feature set. As there are no defined experiment conditions in this dataset, which we can use to aggregate individual ratings into MOS scores, we do not aggregate the dataset at all but instead predict the individual QoE ratings in this use case. This way, we are also able to showcase data-driven QoE modeling in the context of a classification task, for which here the classes would be the five rating categories of the ACR scale ranging from bad (1) to excellent (5) QoE. However, as multi-class classification is often difficult for many existing models and to showcase additional modeling tasks like binary classification, we transform the individual subjective ratings to a binary indicator beforehand to enforce a binary classification scenario. Hereby, a label of 0 indicates low QoE, i.e., a subjective rating of 3 (fair) or lower, and a label of 1 indicates a high QoE, i.e., a subjective rating of 4 (good) or higher. Except for feature scaling, we do not perform any additional preprocessing for the remaining features.

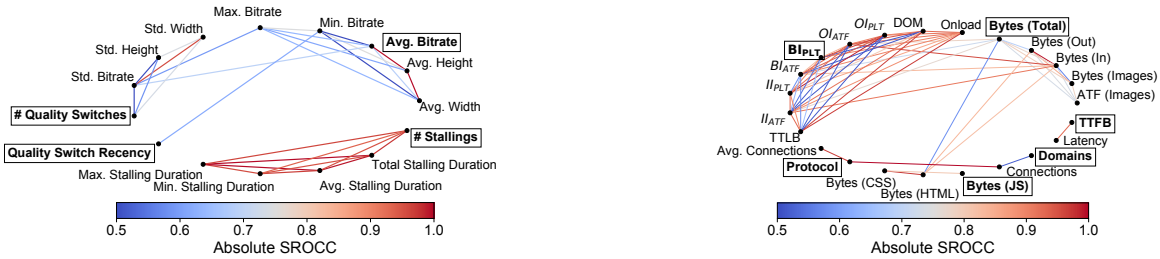
TABLE IV: Utilized features per use case in this work.

	Video Streaming QoE	Web QoE
Feature Set	Expert	Data-Driven
F1	Avg. Bitrate	BI_{PLT}
F2	Initial Delay	Total Bytes
F3	Number of Stalling Events	TTFB
F4	Total Stalling Duration	Number of Domains
F5	Number of Quality Switches	JS Bytes
F6	/	Protocol

Unlike the first use case, the web browsing use case involves both network KPIs and user-centric KQIs, making this scenario a mix of direct and indirect QoE inference (cf. Fig. 1). The network KPIs include BI_{PLT}, total bytes, and TTFB, all metrics directly measurable from the network layer. In contrast, metrics like ATF, OI, or PLT can only be measured by directly analyzing the rendered web page, i.e., working on application layer. In contrast to application-layer QoE modeling, a QoE model purely relying on network KPIs is especially useful for network monitoring by service providers, as it allows to estimate user satisfaction from network-level metrics only.

V. Preprocessing and Expert Modeling based on Domain Knowledge

We refer to other surveys and tutorials on how to preprocess data to setup a valid production ML pipeline, e.g., [66]. In this tutorial, we explicitly do not split our dataset into train and test set since our datasets are too small and generalizability is not our goal here. Instead, we aim to train models providing high



(a) Video Streaming QoE.

(b) Web QoE.

Fig. 3: Graph-based feature selection.

performance and high quality explanations for their internal decision making. Whenever we train classification models, we balance all classes in the datasets by oversampling the minority classes. This step avoids that the model learns to predict the majority class only as this often results in a good performance already with imbalanced datasets. Whenever data normalization for a model is required, e.g., neural models, we apply min-max-normalization to the entire dataset. If not stated otherwise, we always use the mean squared error as loss function for regression tasks and binary cross-entropy as loss function for binary classification tasks. Whenever a model operates on batches, we use a batch size of 32, the default batch size, which works quite well for most ML problems [102]. In the following, we shortly discuss the importance of feature selection and explain how we performed the feature selection for this work. Additionally, we show that the computation of main effects is not sufficient to derive reasonable rules and we discuss how experts with domain knowledge would expect the models to look like.

A. Feature Selection

Selecting the most relevant features from a list of features is easy when you already have domain knowledge about the problem at hand. However, for unknown domains, e.g., for future multimedia applications, it is unclear which features are relevant QoE factors. Thus, we present a data-driven feature selection method based on graph theory, which selects uncorrelated features that are desired for XAI. We will compare the resulting feature set to an expert feature set, which is created based on domain knowledge.

1) Use Case: Video Streaming: First, we perform the feature selection on the video streaming dataset.

Expert: For the expert feature set, we use video QoE domain knowledge derived from the QoE models presented in Section III. In particular, we look at the input parameters of these expert models and select the five most common parameters as features. These expert features include the average bitrate, the initial delay, the number of stalling events, the total stalling duration, and the number of quality switches.

Data-Driven: To select uncorrelated features in a data-driven fashion, we utilize a custom, simple, non-optimized graph-based method similar to [103]. Note that our approach here is not meant as general advice, but only one strategy to find uncorrelated features. Other well-known strategies include Correlation-based Feature Selection [104] or Recursive Feature Elimination [105]. Uncorrelated feature inputs offer several advantages [104]. They reduce redundancy, thus increasing efficiency, and improve interpretability by making it easier to compute feature contributions to model predictions, e.g., with SHapley Additive exPlanations (SHAP) or Local Interpretable Model-Agnostic Explanations (LIME) [106]. This can help avoid misinterpretations. Additionally, multicollinearity [107], leading to inflated coefficient variances in linear or additive models, can be avoided, which enables the application of XAI tools requiring stable model behavior. The mentioned aspects support the intuitive assumption that explanations based on correlated features are less reliable. As shown by Dietz et al. [108], correlated features can indeed reduce interpretability and even lead to lower consensus among competing XAI methods. In contrast to correlation-based techniques, other techniques, such as univariate statistical tests (e.g., selecting the top k features based on their scores) or dimensionality reduction techniques (e.g., Principle Component Analysis), can also be used. However, these methods do not guarantee uncorrelated features. Moreover, dimensionality reduction techniques may decrease interpretability, as the transformed features are no longer directly interpretable.

For our approach, we first compute Spearman's Rank Correlation Coefficient (SROCC) for all features. Using the SROCC, we build a graph by interpreting each feature as a node. Two nodes are connected in the graph if their absolute correlation is higher than a threshold x. In this work, we add an edge between two nodes only if $|SROCC| > 0.5$. Figure 3a depicts the resulting graph for our dataset. Edges between features are colored according to the absolute intensity of the SROCC. We can easily see that all stalling-related features are clustered together and do not share an edge with other features. To find the highly correlated subsets, we then compute all maximal cliques, i.e., fully-meshed subsets of the graph, and sort the cliques by size in a descending fashion. We then iterate through each clique and select the feature with the highest SROCC to the MOS, add the feature to our feature set, and blacklist all other features in the clique. Cliques of size one with $SROCC < 0.2$ are removed from the graph, such as the

initial delay in this case, as they would contribute negligibly to a model prediction. After all cliques have been checked, we obtain the final feature set. For our exemplary dataset, four features are selected as most indicative. These are the average bitrate, the number of stalling events, the number of quality switches, and the recency time of the last quality switch. When comparing the expert and the data-driven feature set, we observe that both feature sets contain the average bitrate, the number of stalling events, and the number of quality switches. Thus, there is a large agreement in both sets, while our data-driven method excludes correlated features, which validates the approach.

2) Use Case: Web Browsing: Next, we perform the feature selection for the web browsing dataset.

Expert: For the expert feature set, we again use web QoE domain knowledge. We select the accessed website and the total downloaded bytes given that it has been shown that these parameters affect web QoE [109]. In several studies, it has also been shown that time-instant metrics like TTLB and time-integral metrics like ByteIndex based on the PLT (BI_{PLT}) are suitable for modeling web QoE [18], [19]. Finally, we select network-related features as it has been shown in literature that these parameters also affect web QoE [17], [100]. This results in six features in total, namely, the accessed website, total downloaded bytes, BI_{PLT}, TTLB, network latency, and network packet loss.

Data-Driven: We replicate the SROCC-based graph for the web QoE data. The resulting graph is depicted in Fig. 3b. Note that to increase intelligibility of the graph, we removed some of the low correlated features from the graph, e.g. most of the network-related features. We observe that six features in total are selected, namely, the amount of total bytes, the TTFB, the number of accessed domains, the amount of bytes of JavaScript objects (JS Bytes), the HTTP protocol version, and the ByteIndex based on PLT (BI_{PLT}).

Compared to the video QoE use case, we observe that the expert and data-driven features do not strongly conform to each other for this use case. However, the feature sets also overlap for the features total downloaded bytes and BI_{PLT}. As we will see later, BI_{PLT} is the most influential feature for modeling web QoE in this dataset. From this perspective, the data-driven approach also correctly identifies the most relevant feature here. We summarize the features along with their identifier (F1-F6) used throughout this work in Table IV. For the video streaming QoE dataset, we utilize the five expert features based on domain knowledge, while for the web QoE dataset, we utilize the six features extracted with our data-driven approach.

B. Main Effects

Before we start to analyze the datasets with XAI, we want to show why it is not sufficient to use simple main effects to obtain valuable insights into the data.

1) Background: A main effect describes the effect of an independent variable, e.g., the number of stalling events, on the target variable, e.g., the MOS, when averaging over all other existing independent variables [110] or their factors/levels, respectively. To obtain these levels and to enhance intelligibility, we discretize the continuous features by binning the features in discrete intervals and averaging all values falling in the corresponding bin. Additionally, we center the MOS beforehand around 0 by subtracting the MOS of the entire dataset. This way, it is easier to interpret the effect of an individual feature on the MOS.

2) Video Streaming: Figure 4a depicts the main effects of the five expert video dataset features on the MOS as well as the interaction effect between bitrate, stalling events and MOS. The gray shades in the background denote the density of the data, i.e., the normalized number of samples available for a specific feature value. Darker shades correspond to higher densities. The y-axis represents how the MOS is affected by individual feature values. We observe that there is a linear positive trend for avg. bitrate and a linear negative trend for number of stalling events visible, but that no trend is visible for initial delay, total stalling duration, and the number of quality switches, as they behave unexpected for some values.

3) Web Browsing: We also investigate the main effects in our web browsing QoE dataset in order to identify obvious feature trends. Figure 4b showcases the main effects for the six data-driven features. The x-axis shows the feature values and the y-axis denotes the effect strength on the binary subjective rating. Grey shades in the background again denote the density of the data. We can easily observe that there is no obvious trend for each of the features visible.

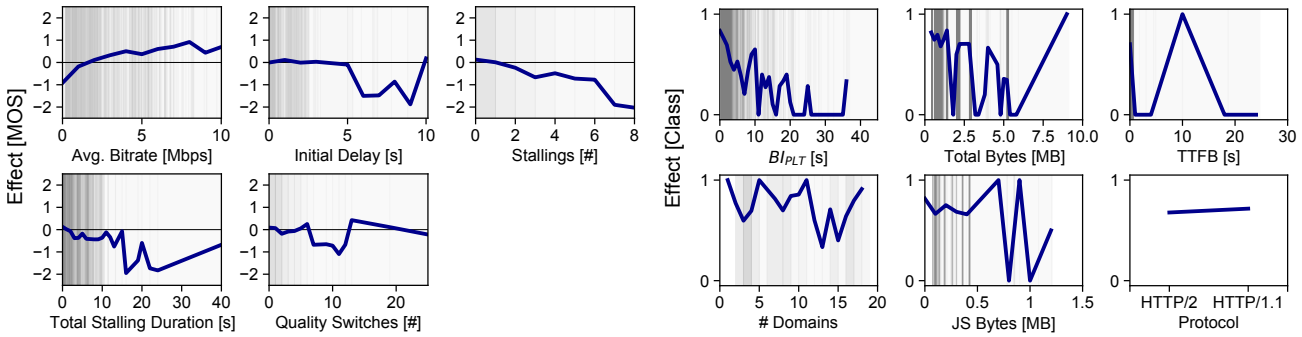
4) Lessons Learned: Main effects are not sufficient to understand which features actually have a strong influence on the MOS, but they only allow to see some simple trends and, in particular, do not consider feature interactions. Additionally, main effects oversimplify relationships as they assume linearity and additivity of effects. They are also prone to return misleading results in the case of unbalanced data [111]. As a consequence, more powerful techniques are required.

C. Expert Modeling

Next, we shortly perform a manual expert modeling, i.e., we provide insights on what kind of model an expert with domain knowledge would expect.

Starting with the video streaming QoE dataset, we would expect that an increasing number of stalling events and an increasing total stalling duration strongly negatively impact the MOS, potentially following the IQX or WQL hypothesis. For the video quality, here expressed with video bitrate as a proxy, we would again assume an exponential or logarithmic relationship based on the IQX and WQL hypothesis, where lower bitrates result in a degradation of the MOS. For the initial delay and also for the number of quality switches, we would expect a linear, slightly negative trend with increasing initial delays and quality switches. However, these features should have a substantially smaller impact on QoE compared to the other features.

For the web QoE dataset, we would expect that solely waiting time related factors, e.g., BI_{PLT}, TTFB, or TTLB,



(a) Video Streaming QoE. (b) Web QoE.

Fig. 4: Main Effects.

affect the MOS and that other features like the number of accessed domains, downloaded HTML bytes, or number of connections would have nearly no impact. Again, we would assume that the waiting time related factors follow the IQX or WQL hypothesis, i.e., longer loading times result in a significantly lower MOS.

Note that each research team has different modeling preferences and that the resulting models would look different. Data-driven QoE modeling with XAI would circumvent this issue by offering objectively optimized models.

VI. QoE Modeling with XAI

We start this tutorial with simple XAI techniques and models and continuously increase the model complexity. We list the interpretable models and post-hoc explanation techniques discussed and utilized in this work in Table V. We begin with linear models and DTs as they are often required to understand later techniques. In general, we keep the level of abstraction as high as possible without going into too many technical or mathematical details. For an in-depth understanding of the individual techniques, we refer the interested reader to the research works listed along with the techniques in Table V.

A. Hyperparameter Optimization

If not stated otherwise, we perform extensive hyperparameter tuning for any model, e.g., the depth of a DT, the number of layers and the number of neurons of the layers in neural networks, or the learning rate. For the tree based models, i.e., DT, RF, and eXtreme Gradient Boosting (XGBoost) we utilize scikit-learn's exhaustive grid search with 5-fold cross validation. With k-fold cross validation, k-1 folds of the training data are used for fitting the model and the remaining fold is used for validating the fitted model. For TabNet, Explainable Boosting Machine (EBM), and Neural Network-based models, we utilize Optuna [126], a framework for performing hyperparameter optimization, as these models cannot be integrated properly into scikit-learn. We use the Tree-Structure Parzen Estimator algorithm [127] to sample a new set of hyperparameters in each training round and utilize a pruner, which stops unpromising training rounds early. Additionally, we apply early stopping to stop training with current set of hyperparameters if there has not been any model improvement observed over the last 10 epochs. Note that sampling the hyperparameters, stopping unpromising training rounds early, and applying early stopping results in significantly lower training times. Each training round consists of 100 epochs and we evaluate at least 100 different hyperparameter combinations. Note that the linear models and the post-hoc explanation techniques do not require any hyperparameter optimization.

B. Evaluation

In the following, we first introduce all XAI models and techniques and describe their explainability and interpretability capabilities in the context of the QoE modeling use cases. We summarize the learned lessons based on our observations for the QoE use cases. Afterwards in Sections VI-J and VI-K, we conduct a quantitative performance and training time evaluation jointly for all XAI models and techniques, feature sets, and use cases and discuss the obtained results. In addition, we also compare expert QoE models, e.g., ITU-T P.1203 for video streaming QoE, to the data-driven models. The corresponding results of these quantitative evaluations are summarized in Tables X, XI, and XII.

C. Linear Models

1) Background: Linear models are inherently interpretable models and are defined by a simple mathematical equation. If we consider a regression scenario, we can apply Linear Regression (LinReg):

$$f(x) = \beta + \sum_{i=1}^{n} \alpha_i \cdot x_i \tag{1}$$

The model is defined by its intercept β and the feature coefficients α_i for each of the features x_i. Linear models belong to the family of additive models, where the individual feature effects are summed up along with a bias to obtain the model output. As we will see throughout this work, additivity is a fundamental property of many interpretable models and post-hoc techniques.

TABLE V: Overview on considered (XAI) techniques and models in this work.

Model/Technique	Scope	Interpretable	Post-Hoc	Explanation	Task
Linear Regression (LinReg) [112]	Global	✓	✗	Additive Functions	Regression
Logistic Regression (LogReg) [113]	Global	✓	✗	Additive Functions	Classification
Decision Tree (DT) [114]	Global/Local	✓	✗	If-Else Rules	Both
Local Interpretable Model-Agnostic Explanations (LIME) [115]	Local	✗	Model-Agnostic	Any Interpretable Model	Both
Anchors [116]	Local	✗	Model-Agnostic	Matching Rules	Classification
SHapley Additive exPlanations (SHAP) [117]	Global/Local	✗	Model-Agnostic	Additive Feature Attribution	Both
TabNet [118]	Local	✓	✗	Feature Attribution	Both
Symbolic Regression (SR) [119]	Global	✓	✗	Equations	Regression
Kolmogorov-Arnold Network (KAN) [120]	Global	✓	✗	Additive Functions	Both
Explainable Boosting Machine (EBM) [121]	Global	✓	✗	Additive Functions	Both
Neural Additive Models (NAM) [122]	Global	✓	✗	Additive Functions	Both
Random Forest (RF) [123]	Black-Box	✗	✗	✗	Both
eXtreme Gradient Boosting (XGBoost) [124]	Black-Box	✗	✗	✗	Both
Multilayer Perceptron (MLP) [125]	Black-Box	✗	✗	✗	Both

LinReg is usually trained with the mean squared error as loss function. There exists a variety of extensions of LinReg, e.g., Ridge regression or Lasso regression, which penalizes the learned coefficients in one way or the other. Lasso regression for example increases sparsity by regularizing the magnitude of the coefficients, and therefore acts as some kind of additional feature selection technique.

For a binary classification scenario, we can use for example Logistic Regression (LogReg) to obtain a linear classification model. The difference between LinReg and LogReg is that the output of the LogReg model f is in the end passed through a sigmoid activation function $g(x) = \frac{1}{1+exp(f(x))}$, which maps the regressed output on the interval $[0, 1]$. Using a threshold, usually 0.5, the model is thus able to classify an instance as belonging to class 1, otherwise as belonging to class 0. To train a LogReg model, the binary crossentropy loss is used. If we consider a multi-class classification scenario, multinomial LogReg should be applied and the categorical crossentropy loss should be used.

Interpretability of linear models is easily obtained by investigating the linear functions projected by the learned feature coefficients and feature values.

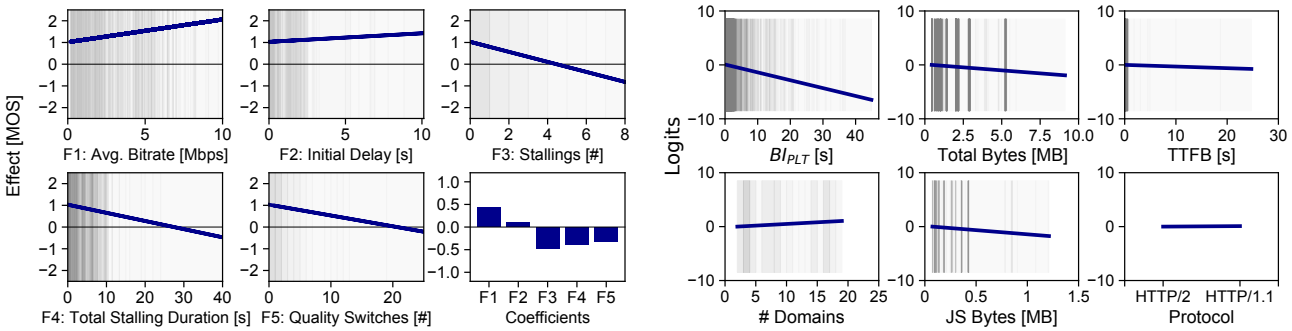
2) Implementation: We use scikit-learn's linear regression model for the video streaming QoE dataset (regression task) and scikit-learn's logistic regression model for the web QoE dataset (classification task). For both tasks, we normalize the datasets to reduce the sensitivity of the parameters and thus stabilize training.

3) Video Streaming: First, we start with the video streaming QoE dataset. Fig. 5a depicts the learned linear functions for each feature as well as the regression coefficients for each feature. As with the main effects, the gray shades in the background denote the density of the data, i.e., the normalized number of samples available for a specific feature value. Darker shades correspond to higher densities. The model considers stalling events, total stalling duration, and quality switches to have a negative impact with increasing values, while an increasing avg. bitrate positively impacts the MOS. Contrary to expectations, for the initial delay, there is also a minor positive impact visible. The feature coefficients in the bottom right sub-plot reveal that the avg. bitrate and the number of stalling events have the highest influence on the output as their coefficients have the greatest absolute value, followed by total stalling duration and the number of quality switches. In contrast, the initial delay does not seem to be that important as its feature coefficient's magnitude is small.

4) Web Browsing: Using LogReg we train a model on the web browsing dataset. The characteristics of the model are depicted in Fig. 5b. Here, we explicitly show the learned linear functions per features and the output of these learned function and not the actual output of the model. This non-normalized output is usually called logits in ML. To obtain the subsequent predictions and the classification probabilities, respectively, the sum of the logits and bias must be passed through the final sigmoid activation function. Due to the definition of the sigmoid activation function, a negative input is going to return a probability below 0.5 and therefore would be labeled as 0. When considering the figure, it becomes quickly evident that the feature BI_{PLT} dominates all other features in terms of total impact and that an increasing BI_{PLT} substantially decreases the probability to obtain a label of 1, i.e., a high QoE rating of 4 or 5.

5) Lessons Learned: Linear models are easy to implement and are quickly trained. They have the benefit of being easily understandable due to the feature coefficients and in contrast to main effects, they allow to model the relationship between the dependent variable and multiple independent variables. On the video streaming and web QoE datasets, the linear models also performed reasonably as the obtained feature importances are mostly in line with research. In particular, the linear models correctly identified stalling events to have a severe negative impact on QoE for the video streaming use case, and they also correctly identified BI_{PLT} as most important feature for web QoE. However, from research it is also known that the linear models are not correct here [15], [80]. While linear models remain a valuable benchmark for initial data exploration and analysis to derive some simple trends, it is possible that we might miss more important non-linear relationships in the data due to their limitations. Therefore, linear models may not be the best approach for all kinds of data and non-linear models might be required.

This article has been accepted for publication in IEEE Communications Surveys & Tutorials. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/COMST.2025.3583227



(a) Video Streaming QoE: Linear Regression.

(b) Web QoE: Logistic Regression.

Fig. 5: Linear models.

D. Decision Tree

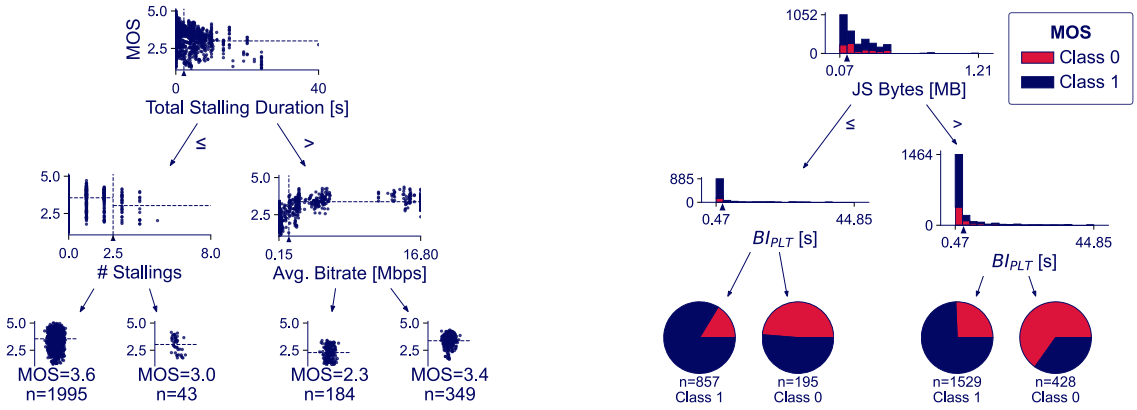
1) Background: Due to their simplicity, Decision Trees (DTs) are nowadays very popular as interpretable models. Well-known DT algorithms are ID3 [128], C4.5 [129], and CART [114]. A DT is learned by recursively splitting the training dataset into smaller subsets until no further splits are possible. The algorithm starts by taking the current dataset and searching for a feature and feature value which results in the two purest subsets. To compute purity, impurity metrics can be used, e.g., Gini impurity, entropy, or gain ratio. The algorithm then splits the training dataset based on the feature and the feature value and produces the new subsets S_l and S_r. If the subset is already pure or if there is no split possible, the algorithm returns the current subset with a classification or regression label as leaf. This training process now repeats for the newly generated subsets S_l and S_r until all subsets return a leaf. The loss function minimized during training is thus the impurity sum of the subsets weighted by the number of instances. Due to the splits of the data based on feature values, the DT can be interpret in an if-else fashion later. Individual decisions can be traced back by traversing the tree from the root to the corresponding leaf node.

2) Implementation: We use scikit-learn's DT classifier and regressor based on the CART algorithm. We optimize the model parameters maximum tree depth, splitting criterion, and the number of maximally used features with three levels each during 5-fold cross validation. The target metrics during cross-fold validation are F1-score for classification tasks and the mean squared error for regression tasks. In the following, we show DTs fitted on a maximum depth of 2 to increase intelligibility of the model internals. For the actual performance evaluations later on, we also consider trees of higher depths and even unbounded trees. Entire trees can also be quickly visualized. In this work, we visualize the individual DTs using the Python library *dtreeviz*.

3) Video Streaming: Figure 6a shows the DT of depth 2 learned on the video streaming QoE dataset. On each tree level, the figure shows at which feature value the model performed the split (triangle), directly explaining the internals of the tree. In the first split, the model splits the dataset by a total stalling duration of 2.26 seconds, before it uses the number of stalling events (2.5) and the avg. bitrate (0.15) to perform the next splits. In the end, we for example see that the model predicts a MOS of 3.6 if the instance's total stalling duration is below 2.26 and less than 2.5 stalling events occurred during the video session. These model decisions make sense from the perspective of a domain expert, as both lower stalling events and higher avg. bitrate should generally lead to better QoE.

4) Web Browsing: For the binary classification of the web browsing dataset, we also depict a DT fitted for a maximum depth of 2 in Fig. 6b. We can quickly see that for a classification scenario *dtreeviz* indicates how many instances of a bin belong to class 0 (red) and class 1 (blue). On the first layer, the tree uses the feature $JSBytes$ and a value of 0.07 to split the dataset into new subsets. The first subset containing the instances with $JSBytes < 0.07$ consists mostly of instances with class 1, i.e., a QoE score of 4 or better, while the other subset is more balanced with respect to the classes. In the next splits, both times BI_{PLT} and a value of 0.47 is used for splitting. The instances with a BI_{PLT} lower than 0.47 are then assigned to leaves reflecting class 1. This split is reasonable and in alignment with domain knowledge as well as with the previous linear models in that the loading time influences the class label the most.

5) Lessons Learned: In general, DTs are also trained quickly, even with extensive hyperparameter optimization, making them an efficient choice for many tasks. They are easily interpretable and provide tools for visualizing model structure and tracing prediction paths, offering clear insights into model decisions.. However, if a DT is not properly regularized, e.g., by limiting tree depth, the model is prone to overfitting, leading to poor performance on unseen data. Moreover, interpretability decreases with increasing tree depth as global explanations are no longer easy to derive. For example, the best DT for the web QoE dataset had a depth of 23, making it difficult to interpret it effectively. The main drawback, however, is that a DT usually underperforms compared to more complex models. Similar to linear models, though, DTs are valuable tools for initial data exploration and analysis. Our evaluation in the video streaming and web QoE use cases also



(a) Video Streaming QoE: Decision Tree.

(b) Web QoE: Decision Tree.

Fig. 6: Decision Trees.

TABLE VI: Overview of LIME.

Step	Action
1	Select instance x_i to explain
2	Perturb instance x_i to generate new samples $\widetilde{X}$
3	Use trained black-box model f to predict $\widetilde{y}$ from $\widetilde{X}$
4	Create a new training dataset consisting of $\widetilde{X}$ and $\widetilde{y}$
5	Weight the dataset samples with proximity function π
6	Train interpretable white-box model g, e.g., linear model, on weighted dataset
7	Analyze explanations returned by interpretable model g

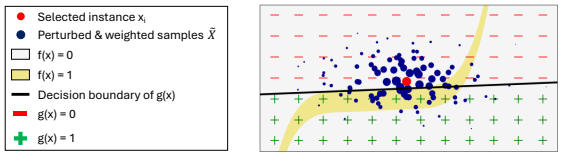


Fig. 7: An illustrative example for LIME.

showed that they align well with domain knowledge, where they were also able to identify the critical features. Nevertheless, it is important to note that DTs serve as building blocks for ensemble-based tree models, e.g., RF and XGBoost [124], a state-of-the-art model for tabular data. While these models are more black-box, they often show excellent performance. Nevertheless, it is possible to interpret these ensemble-based tree models by using post-hoc explanation techniques.

E. Post-Hoc Explainer

1) Background: Post-hoc explainers [43] are utilized to explain various black-box models, e.g., neural networks or tree-based ensembles, after they have been trained. In the following, we cover a selected set of perturbation-based post-hoc explanation techniques, namely, LIME [115], Anchors [116], and SHAP [117]. Both Anchors and SHAP build on top of LIME, the oldest technique of the three. Therefore, we start with introducing LIME, followed by Anchors and SHAP.

a) LIME: The idea behind Local Interpretable Model-Agnostic Explanations (LIME) [115] is to train a white-box model, which approximates the predictions of the black-box model for a given instance. A model trained to approximate another model is also called surrogate model. Table VI depicts the general workflow of LIME and a specific classification example is illustrated in Fig. 7. In the example, the red dot denotes the instance x_i, positioned relative to its feature inputs on the axes, for which we want to understand how the black-box model f derived its decision. The decision boundary of f is illustrated by the gray and yellow background, indicating predictions for class 0 and class 1, respectively. In the first step, LIME generates a new dataset $\widetilde{X}$ by applying random perturbations to x_i (blue points). The kind of perturbation applied always depends on the considered data type: for images, random pixels are turned off; for text, individual words are removed; and for tabular data, feature values are sampled from a normal distribution based on the feature's mean and standard deviation. Next, LIME uses the black-box model f to obtain predictions $\widetilde{y}$ for the perturbed dataset $\widetilde{X}$. Using $\widetilde{X}$ and $\widetilde{y}$, LIME trains an interpretable, surrogate model g, such as a linear model or a DT, to predict $\widetilde{y}$ from $\widetilde{X}$. During training, samples in $\widetilde{X}$ are additionally weighted according to a proximity function π, giving higher weights to samples closer to x_i (as illustrated by the size of the blue points). The surrogate model g, here a linear model, can then be interpreted to understand how the black-box model f actually derived its decision. This is achieved by analyzing g's linear decision boundary (black line) and predictions (shown as - and +). In this example, we learn that the distance between classes is relatively small for instance x_i, indicating that the instance lies close to the decision boundary of f.

b) Anchors: Anchors (or scoped rules) [116] is from the same authors as LIME. The idea behind Anchors is to find rules, which *Anchor* the prediction of a black-box classification model, i.e., we obtain a rule which fits many instances and is not easily changed by other feature values. Such rules consist of predicates, which can be interpreted in an if-else fashion, e.g., *# stallings* $= 0$ and *avg. bitrate* > 1 *Mbps*.

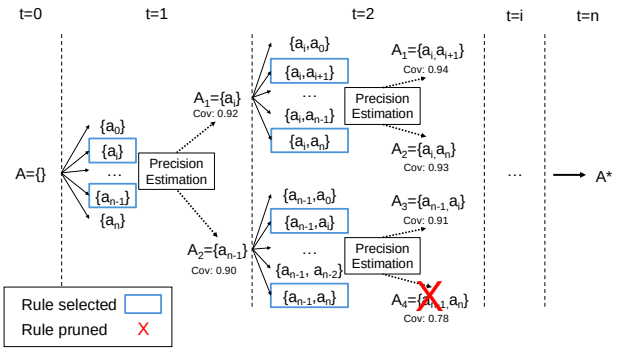


Fig. 8: Anchors.

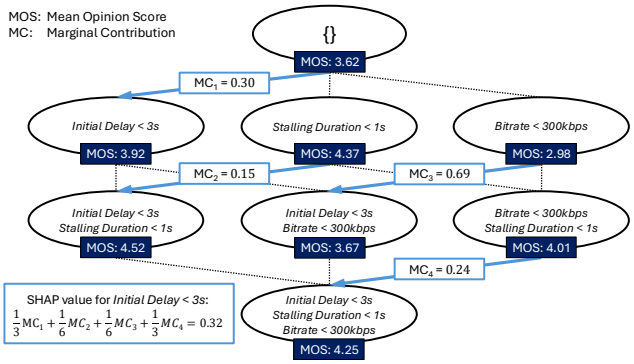


Fig. 9: An example for the SHAP computation.

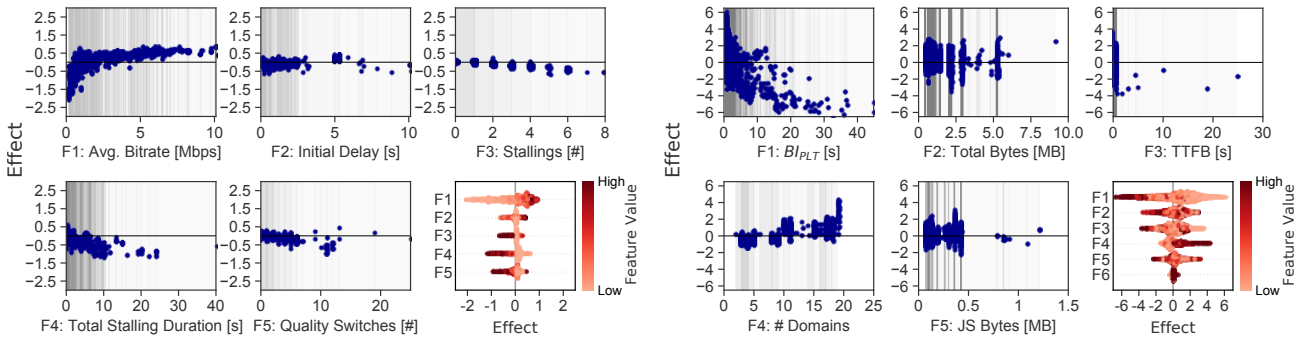
We consider the conditional distribution of perturbed samples for an instance, where the considered predicates remain fixed. A predicate set is considered an Anchor if, for all perturbed samples in the distribution, the same prediction (e.g., *MOS* $>$ *4*) occurs with high probability, even if not all instances adhere to the predicate set. To quantify the quality of an Anchor, Anchors use both precision and coverage. Precision simply states how often the Anchor actually matches, while coverage states how many instances in the perturbed dataset are covered by the Anchor. An Anchor with a precision of 0.8 and a coverage of 0.15 thus states that 15% of the instances in the perturbed dataset are covered by this Anchor and in 80% of these instances the prediction of the Anchor is correct.

However, finding the best Anchors is challenging as there exists an exponential number of possible Anchors as the number of predicates increases. By considering both precision and coverage, this task can be framed as a combinatorial optimization problem. To address this, Anchors leverage beam search and reinforcement learning. Figure 8 depicts the general workflow of identifying Anchors. The search for an Anchor begins with an empty rule set. At each step, new predicates are added to active sets, and their precision is evaluated. As the precision computation is intractable, reinforcement learning is used to approximate it. To avoid to select only the current best rule to continue with and miss out on better rules later, beam search identifies the best set of rules based on computed precisions. If multiple rules have the same precision, those with higher coverage are preferred. Once the search is complete, the best Anchor is returned. Anchors provides interpretability by revealing the features the model relies on for a given prediction, along with relevant feature ranges.

c) SHAP: SHapley Additive exPlanations (SHAP) [117] is nowadays one of the most popular post-hoc explainers. The authors of SHAP identified that several already existing techniques like LIME, DeepLift [130], Layerwise Relevance Propagation [131], and Shapley values [132] also possess the property of additivity. They also showed that these additive feature attribution methods have a unique solution with three desirable properties, but that existing approaches did not fulfill these. These properties are local accuracy, i.e., the explanation model must match the output of the original model, missingness, i.e., missing features in the input should have no impact on the explanation, and consistency, i.e., the explanation value must consistently change with the contribution (e.g., an increase in the attribution results in an increase in the explanation value). Interestingly, Shapley values from game theory provide this unique solution.

SHAP computes the contribution of a feature (its SHAP value) by comparing the model's predictions when including and excluding a feature across all possible feature sets. Figure 9 illustrates this process with three features: *Initial Delay<3s*, *Stalling Duration<1s*, and *Bitrate<300kbps*. With three features, there exist $2^3 = 8$ possible feature subsets. At the first level, we start with the empty subset, where none of the features are present, and where the black-box model predicts a MOS of 3.62, the average prediction across all instances. At the second level, subsets of size 1 are created by adding the individual features and recomputing the MOS for all subsets. This process is repeated for the third and the fourth level.

Next, we show how the SHAP value for the feature *initial delay<3s* is computed. The process begins by first calculating the marginal contributions when the feature is added to a feature subset. A marginal contribution is the difference in the model's output (MOS) when the feature is included versus excluded from feature subsets. To identify the marginal contributions, we look for edges in the figure where the target feature is added to a subset (identified ones are marked with blue arrows). For example, when moving from the empty subset to the subset including only the target feature, the marginal contribution is $3.92 - 3.62 = 0.30$. Each marginal contribution is additionally weighted to account for the number of possible feature combinations. The weights are derived by counting the (hypothetical) number of edges between the different levels in the figure. We have three possibilities to derive feature subsets of size 1 when starting from the empty subset. Subsequently, the weight for each of these edges is $\frac{1}{3}$. Starting from the feature subsets of size 1, we have six possibilities in total to end up in the respective feature subsets of size 2. Consequently, the weight is $\frac{1}{6}$ for each edge. For



(a) Video Streaming QoE.

(b) Web QoE.

Fig. 10: SHAP values.

the last feature subset, there are again three possibilities, so the weights are $\frac{1}{3}$. By summing all weighted marginal contributions, we obtain the SHAP value for the target feature. In this example, the SHAP value of *Initial Delay<3s* is 0.32, meaning the MOS increases by 0.32 when the initial delay is kept below 3 seconds. Similarly, the SHAP values for *stalling duration<1s* (0.72) and *bitrate<300kbps* (-0.41) can be computed. Here, the SHAP values indicate that stalling durations below 1 second have a strong positive impact on the MOS, while low bitrates significantly decrease it.

Computing SHAP values, therefore, requires multiple model predictions for each subset, both with and without the feature under consideration. This computation can be expensive, especially for larger models or when dealing with a high number of features. One reason for the popularity of SHAP nowadays is thus that the authors found ways to significantly speed up the computation. The authors proposed Kernel SHAP, which approximates Shapley values by adapting LIME. They identified the optimal settings of LIME, i.e., loss function and weighting function π and were thus able to approximate Shapley values with a linear model g. The coefficients of g then correspond to the SHAP values ϕ_i and the intercept of g to ϕ_0. The authors also proposed other approximations for other use cases, e.g., Deep SHAP. Another reason for the popularity of SHAP is also Tree SHAP [133], which efficiently computes Shapley values by traversing the grown tree and averaging the marginal contributions of the nodes.

2) Implementation: For the computation of SHAP values we use the Python package *shap* [134]. As black-box model, we use XGBoost, which we first train on the datasets of the use cases each. As we apply SHAP on top of XGBoost, we use the provided Tree SHAP modules. For the computation of the Anchors, we use the Python package *anchor-exp* [135] and apply the tabular explainer to the web QoE dataset.

3) Video Streaming: Next, we explain the XGBoost model's internals as described by SHAP values. Figure 10a shows how MOS or SHAP values, respectively, are affected by different feature values. The obtained SHAP value on the y-axis represents how a single instance with a given feature value differs from the average of the entire dataset. For example, considering F1, stimuli with an average bitrate below 1 Mbps show a MOS that is around 1-2 points lower on average than the average MOS of all stimuli. Although SHAP value plots resemble traditional main effect plots, they have the advantage of implicitly accounting for feature interactions.

TABLE VII: Exemplary Anchors for web QoE dataset.

If	Predict	Precision	Coverage
JS Bytes ≤ 0.3	1	0.98	0.76
$BI_{PLT} > 1.02$	1	0.99	0.75
TTFB > 0.29	1	0.99	0.25
$BI_{PLT} \leq 1.02$, Protocol=HTTP/2	0	0.11	0.04

We notice that the SHAP values of the average bitrate show a strong monotonic trend, thus indicating the importance of this QoE factor. When we consider the initial delay (F2), we observe SHAP values centered around 0, i.e., negligible impacts on MOS. For the number of stalling events (F3) and the number of quality switches (F5), there are also only minor negative linear trends visible. In contrast, the total stalling duration (F4) has a stronger negative impact with increasing stalling duration, which comes as expected. On the bottom right, a summary over the SHAP values per feature is shown. The figure shows the effect of the individual SHAP values for the features F1-F5 (indicated on the y-axis) on the MOS (indicated on the x-axis). The darker the color, the higher the feature value. This once again clearly confirms the previously observed trends. We remark that the SHAP values in our figures show some variance, e.g., cf. F4 between 10 and 20 seconds, and thus, are not very accurate and difficult to interpret in mathematical terms.

4) Web Browsing: Next, we aim to explain the internals of an XGBoost model trained on the web browsing dataset. We show the obtained SHAP values in Fig. 10b. We can observe that BI_{PLT} (F1) shows the easiest to interpret and a very similar trend to the trend derived by the logistic regression model. While total bytes (F2), TTFB (F3), and JS bytes (F5) do not show a clear trend, the feature protocol (F6) has no effect at all, and was thus omitted from the plot. Only the feature number of domains (F4) shows a slight positive trend

for an increasing number of domains, i.e., the more domains have been accessed by a website the better the QoE. Accessing more domains potentially results in downloading more CSS style files, images, or other assets and may therefore be more appealing to the end-user even if loading times are as slow as for other websites.

As we consider a binary classification scenario here, we can also apply Anchors to the learned models. Some exemplary rules obtained after applying Anchors to the XGBoost model are listed in Table VII. We can see that the simple rules $JSBytes \leq 0.3$ and $BI_{PLT} > 1.02$ have a high precision of 98% and 99%, and a coverage of 76% and 75%, respectively, and always predict a MOS of 4 or 5. While the first rule seems reasonable, the second rule with BI_{PLT} is counter-intuitive and contradicts our domain knowledge, since higher page load times should not result in a higher QoE. The results are thus misleading as high precision and coverage are suggested for an instance, but in reality the metrics concern only the perturbation space, i.e., artificial data, not the original data space. If we revisit the original data, we observe that only a small number of instances is actually close to this very limited value range. If we consider other rules, we observe that for the majority of data, Anchors could not find any rules with high precision, where the original data range is also large, e.g., rule 4. As a consequence, this rule has limited explanatory power.

5) Lessons Learned: Post-hoc explanation methods, such as LIME, Anchors, and SHAP provide valuable insights into black-box models by reporting feature importance and decision boundaries. The usage of such techniques is particularly reasonable when explanations are required for an already trained black-model at a later stage. However, these techniques also introduce additional computation overhead, especially with SHAP, where the number of required model predictions grows exponentially with the number of input features and samples. While SHAP provided explanations in line with previous models and offers tools to easily generate visualizations through its API, it comes at the cost of increased computation time. In contrast, Anchors often produced rules that were not only difficult to interpret but also misleading likely due to the fact that Anchors works on a perturbed data space rather than the actual dataset. Given these limitations, we argue that inherently explainable models may be the better choice for data-driven QoE modeling, particularly for scenarios, where modeling pipelines are not yet in place.

F. TabNet

1) Background: Usually, tree-based models like XGBoost [118] clearly outperform neural networks on tabular data. TabNet was introduced as the first neural network architecture able to compete on tabular data with XGBoost. Its sequential attention mechanism enables step-wise feature selection through masking, providing explainability by highlighting the features influencing each prediction. Figure 11 depicts the internals of TabNet on an abstract level. During each decision step, two black-box functions, i.e., the neural subnetworks Feature Transformer and Attention Transformer,

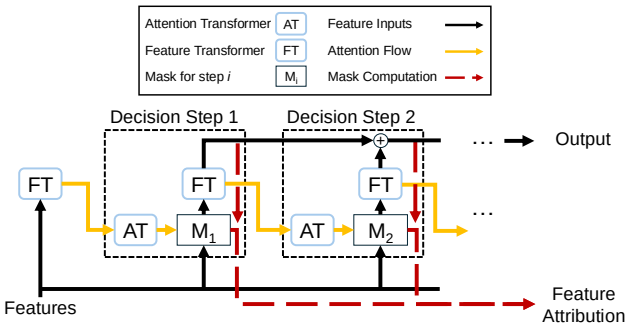
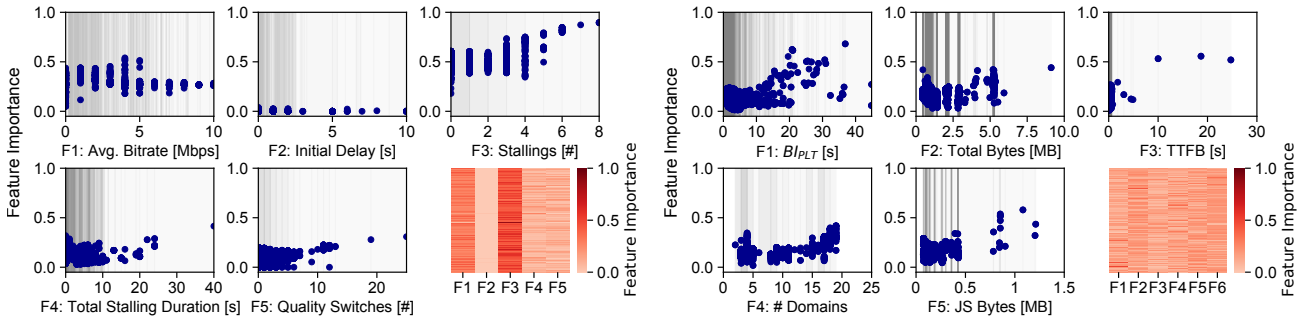


Fig. 11: TabNet.

are used. The Feature Transformer transforms the masked features of the current step and splits the output into a decision step output (black line on top) and an additional information output (yellow line). In each decision step, the Attention Transformer uses this additional information output of Feature Transformer to compute a mask M_i, which determines which features should be considered in the current step. The original input features are thus multiplied with the mask and passed to the Feature Transformer again. The final model output is then obtained by summing the decision step outputs and passing the obtained logits through a final activation layer, e.g., sigmoid activation. All explainability of TabNet stems from the learning of the masks for each decision step. By multiplying the decision step output with the learned mask of each corresponding step and summing over these outputs, TabNet obtains a feature attribution map in the end, which can be used to peek into the model.

2) Implementation: A PyTorch implementation of TabNet is provided in the Python package *pytorch-tabnet* [136]. During hyperparameter optimization, we optimize the learning rate, the number of decision steps, the number of neurons used for Feature Transformer and Attention Transformer, and the relaxation factor, which states the coefficient for feature reusage in the masks. One advantage of TabNet is that no normalization is required with this model, so we explicitly do not normalize the data here.

3) Video Streaming: We depict the feature attribution map learned by TabNet on the video streaming QoE dataset in the bottom right of Figure 12a. The x-axis denotes the features and the y-axis depicts each instance of the training dataset in form of a horizontal bar. These horizontal bars are colored feature-wise according to the feature importance of the individual features of the instances. The darker the line, the more important TabNet considered the feature for its prediction. We can observe that TabNet focuses mainly on the features avg. bitrate (F1) and the number of stalling events (F3) for all instances (dark red vertical bars), while it deems the other three features as significantly less important. We additionally illustrate in the other figures how the feature importance of TabNet varies when the individual feature values increase or decrease, respectively. In alignment with the heatmap, we see



(a) Video Streaming QoE.

(b) Web QoE.

Fig. 12: TabNet.

that the feature importance for the avg. bitrate (F1) and the number of stalling events (F3) is close or above 0.5, while the other features show values close to 0, in particular the initial delay (F2). While the feature importance for the avg. bitrate stays similar with increasing bitrates, it increases for the number of stalling events significantly. For six or more stalling events, TabNet uses more than 90% of its feature importance for the number of stalling events. As a consequence, we realize that the model understood from the data that many stalling events are a sufficient indicator for a low QoE.

4) Web Browsing: The TabNet model explanations obtained for the web browsing dataset are shown in Fig. 12b. In contrast to the video streaming QoE dataset, the feature importance heatmap looks less conclusive. The investigation of the feature importance variation shows some more interesting insights. An increasing feature value of BI_{PLT} (F1) also results in an increased feature importance, indicating that the model understood that higher loading times result in a lower probability of obtaining a QoE of 4 or 5. Also, larger web pages with respect to the total bytes (see F2) result in a higher feature importance for this feature.

5) Lessons Learned: Compared to previous models, TabNet's training process, along with hyperparameter tuning, is more computationally intensive and time-consuming (cf. Section VI-K). TabNet provides interpretability by providing both local and global explanations through feature attribution maps generated by the attention mechanism, offering insights into the model's decision-making process. An additional feature of TabNet, not discussed yet, is the fact that it supports self-supervised pre-training. For this purpose, TabNet masks specific feature entries and attempts to reconstruct these masked feature entries using the context provided by the unmasked feature entries. This pre-training approach reduces the dependency on large datasets and allows to achieve comparable performance with fewer samples. Despite the resource demands, the model offers good performance and data efficiency, and subsequently makes it a valuable model for data-driven QoE modeling where feature importance and the ability to work with limited labeled data are crucial. While TabNet's heatmaps provide valuable insights in terms of feature importance, they are often less interpretable than explicit rules, which are easier for stakeholders and non-experts to understand.

G. Symbolic Regression

1) Background: As the name suggests, Symbolic Regression (SR) works only for regression tasks. SR is usually based on the principles of evolutionary algorithms [119], [137]. Its goal is to derive a symbolic equation, consisting of an arbitrary number of independent variables and arithmetic operators, which best fits the data with respect to the dependent variable. The term best hereby refers to a trade-off between the degree of complexity, i.e., the amount of independent variables and operators required for the equation, and the obtained loss using an arbitrary loss function, e.g., Mean Squared Error (MSE). Thus, learning such equations is a multi-objective optimization problem where prediction error and equation complexity should be kept as low as possible. The final symbolic equations are hopefully understandable to humans and thus by design interpretable. However, the degree of interpretability decreases with increasing equation complexity.

Figure 13 depicts the general, simplified process of learning symbolic equations, here in detail with PySR [137]. First of all, there exists a population of individuals, here symbolic equations, e.g., $-0.54 \cdot ln(t) + 7.90$ or $t^2 + 0.3$. These symbolic equations are internally represented by trees as shown in the lower left of the figure. Evolutionary algorithms work iteratively until a criterion is met or the total number of rounds has been exceeded. Each round looks as follows: First, a random subset of the current population is sampled. Then, a tournament is executed where the performance of the selected samples is evaluated with a fitness (loss) function. The fittest expressions are then copied to the next stage, the offspring generation. Here, random mutation operators are applied to the expressions. PySR implements simplification, constants optimization, a simple mutation, and crossover. During simplification, PySR tries to reduce equation complexity by removing redundant parameters. In this exemplary case, nothing changes in the equation. With constants optimization, PySR tries to improve the equations by finding better constant values resulting in a higher fitness. During mutation, a random operator is changed. In the example, the operator $+$ is flipped to $-$, thereby creating a new expression. For crossover, two

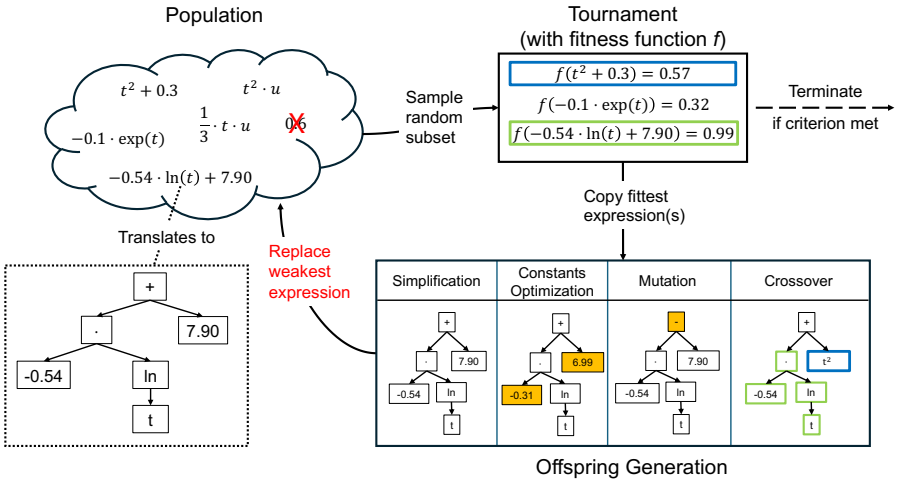


Fig. 13: Symbolic Regression.

TABLE VIII: Best performing rules with symbolic regression.

		Equation	MSE	Complexity
Video	1	0.604	0.036	1
	2	$tanh(F1 + 0.552)$	0.028	4
	3	$tanh(0.518 + 1.336 \cdot F1)$	0.028	6
	4	$0.722 - 0.397 \cdot exp(-16.861 \cdot F1)$	0.021	8
	5	$0.353 + (tanh(-11.144 \cdot F1) \cdot (F4 - 0.407))$	0.018	10
Web	1	0.4999	0.250	1
	2	$(F1 - 0.773)^2$	0.238	4
	3	$0.717 \cdot exp(-8.745 \cdot F1)$	0.217	6
	4	$exp(F4 - 9.277) \cdot (F1 + 0.037)$	0.216	8
	5	$tanh(F4 + 0.677) \cdot exp(-10.411 \cdot F1)$	0.215	9
	6	$0.213 \cdot F4 + exp(-13.979 \cdot F1) - 0.370$	0.212	10

expressions are randomly combined to a new one. In the example in the figure, $t^2 + 0.3$ and $-0.54 \cdot ln(t) + 7.90$ are merged to $-0.54 \cdot ln(t) + t^2$ by merging the first terms of both expressions. The generated offsprings are finally added to the population and the weakest expressions are removed from population before a new tournament phase starts. In PySR, complexity is defined as the number of nodes in an expression tree regardless of content. The complexity of constants and operators can be arbitrarily defined (default is 1 for all operators, constants, and instances of variables).

2) Implementation: In this tutorial, we use PySR [137] to derive symbolic equations. We allow the additive and multiplicative binary operators as well as the unary operators exp, log, $tanh$, and $square$ only.

3) Video Streaming: We train the model on the video streaming QoE dataset for 10,000 evaluations. The five best performing rules for the expert feature set obtained after training PySR on the video streaming QoE dataset are shown in Table VIII. We see that rule 1 is very simple and solely consists of a constant value, the normalized MOS of the entire dataset. The rules 2 and 3 both use the tanh function, the avg. bitrate (F1) as feature, and some factors and constant values. While the rules 1-3 have low complexity and are thus easily understandable, rules 4 and 5 have higher complexity and are thus less easy to interpret, even though they better fit the dataset. Interestingly, rule 5 considers the total stalling duration (F4) as most important feature after avg. bitrate (F1). The most reasonable rules to use in practice would be likely rules 2 and 3 as they are easy to interpret.

4) Web Browsing: For the web browsing QoE dataset, we reformulate the original task of binary classification to a regression task, i.e., we use the labels of 0 and 1 as target variable, and again evaluate 10,000 evaluations. The six best performing rules for the data-driven feature set obtained after training PySR on the web browsing QoE dataset are shown in Table VIII. Again, we have a constant value of 0.4999, which results in a model predicting class 0 only (after rounding 0.4999 we obtain 0). Rule 2 is a quadratic function and rule 3 is an exponential function both based on BI_{PLT} (F1). The remaining rules (4-6) all utilize both BI_{PLT} (F1) and the number of domains (F4) and relate them using linear, exponential, or tanh functions. For these three rules complexity is also again high (8+) and they are thus more difficult to interpret. Domain knowledge tells us that the most likely rule to really match is rule 3 where the loading time BI_{PLT} is mapped according to the IQX hypothesis.

5) Lessons Learned: Using SR to obtain valuable symbolic equations is a complex task as extensive hyperparameter tuning is required. While its algorithmic flexibility broadens its applicability to diverse domains, ensuring proper penalties for unary operators (e.g., logarithmic or exponential function), binary operators (e.g., multiplication or addition), and other operators is critical to achieve an excellent trade-off between performance (loss) and complexity. Moreover, overly nested expressions must be avoided to preserve interpretability. In our experiments, only one valuable rule (rule 3) emerged. This rule, characterized by a few operators and features only, was easy to interpret and thus practical for real-world deployments. However, the fact that only a single reasonable

rule could be identified, highlights the difficulties of acquiring meaningful and practical equations with SR. Further, none of the rules generated for the video streaming QoE use case effectively captured and modeled the impact of stalling events, a key video streaming QoE influence factor. These limitations generally question SR's suitability for QoE modeling.

H. Kolmogorov-Arnold-Network

1) Background: The Kolmogorov-Arnold Network (KAN) [120] is the most recently proposed model in this work and is considered as an interpretable, parameter-efficient alternative to Multilayer Perceptrons (MLPs) by the authors. It is based on the Kolmogorov-Arnold-Theorem, which states that any multivariate continuous function f can be represented using a composition of univariate continuous functions of a single variable and the binary operation of addition.

Like MLPs, KANs also build a graph or network and thus consist of nodes and edges. In contrast to MLPs, where the nodes have learnable parameters and where the edges activate the node outputs using fixed non-linear activation functions, KANs perform the binary operation of addition on the nodes and they learn the activation functions on the edges. Using this approach, the authors were able to generalize the theorem to the machine learning domain. They also introduce the notion of KAN layers, and show that it is possible to increase the depth of the network by stacking multiple KAN layers. The 1D activation functions are learned by approximating the true function with piece-wise polynomial functions (b-splines) in arbitrary grid intervals (similar to bins). These grid intervals depend on a hyperparameter g, which controls whether activation functions are allowed to be more fine-grained (a large g and small intervals) or more coarse-grained (a small g and large intervals).

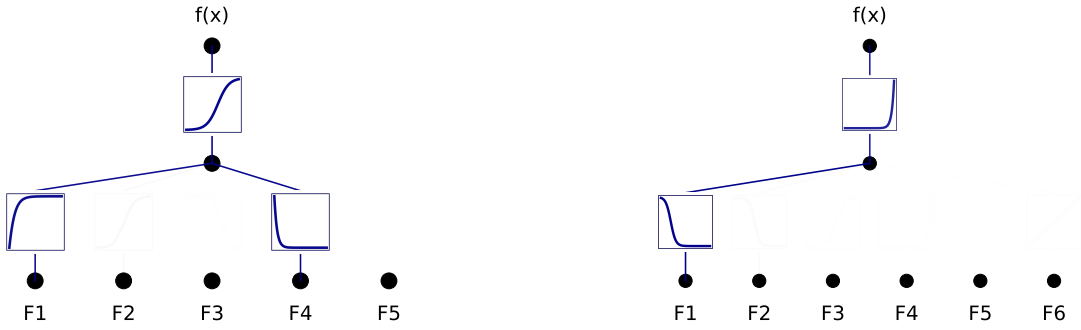
Interpretability of KAN is manifested by the learned activation functions, which clearly describe how the inputs are activated, and from the simple binary operation of addition. However, as with DTs, it is difficult to interpret the model, when the number of layers increases. To ease this understanding, KAN also provides sparsity regularization and pruning techniques to keep the networks as compact as possible. The second aspect of interpretability comes from the fact that KAN can be used for SR. KAN allows to fit the learned activation functions to symbolic functions, e.g., log, exp, x^2, and then use these functions to generate a SR formula.

2) Implementation: For the implementation of KAN, we use the official Python package *pykan* [138]. During training we put more focus on interpretability and less focus on performance. As recommended by the authors of KAN, we thus introduce sparsity by increasing regularization during training and start the training with a small g and increase g every 50 epochs. After training, we prune the network and fix the symbolic functions, i.e., we fit each activation function to a pool of allowed functions. This pool consists of the linear identity function (x), the squared function (x^2), the exponential function ($exp(x)$), the logarithmic function ($log(x)$), and the hyperbolic tangent function ($tanh(x)$). We formulate both the video streaming and web QoE dataset as regression problem. For the web QoE dataset, we observed a better performance with the regression formulation (one output, a threshold of 0.5 determines the class as with logistic regression, and the model is trained with the mean squared error) than with the classification formulation (two outputs, and the model is trained with cross-entropy loss).

3) Video Streaming: We depict the resulting KAN for the video streaming QoE dataset in Fig. 14a. In the first layer on the bottom, the five expert features F1-F5 (avg. bitrate, initial delay, number of stalling events, total stalling duration, and number of quality switches) are fed into the network. While F1 and F4 have an outgoing connection to the node of the second layer, F2, F3, and F5 do not have such connection. This states that these features have been pruned as KAN determined that they are less relevant to the output. On the edges between the first layer and the second layer, the fitted activation functions for F1 and F4 are shown. The activation function for the average bitrate has been fitted with the $tanh$ function, while the activation function for the total stalling duration has been fitted with the log function. Considering the activation function of the avg. bitrate (F1) now, we see that the activation starts from 0 for low bitrates and increases quickly for higher bitrates until the plateau of 1 is reached. This is in line with previous and following models and indicates that there is no gain in QoE possible after a specific bitrate was reached. For the total stalling duration (F4), we can see the opposite behavior, where a low total stalling duration results in high activated outputs, thus a higher QoE, but that there is a rapid decrease in the activated output as soon as the total stalling duration increases until the plateau of 0 is reached. Considering the final activation function, i.e., the activation function which takes the sum of the outputs of F1 and F4 as inputs, we observe that the resulting outputs, i.e., MOS, follow a slight linear or sigmoid function. This means that higher activated outputs of F1 and F4 also lead to a higher MOS and vice versa.

4) Web Browsing: The learned model for the web QoE dataset is shown in Fig. 14b. We can see that from the six used features on the bottom only the feature F1 (BI_{PLT}) remains after pruning. The resulting activation function of F1 could be mapped to the $tanh$ function. If we consider the activation function of F1, we can see a high output for low BI_{PLT} and that an increase of BI_{PLT} results in lower outputs and subsequently a prediction of the bad QoE class. We can see that the final activation function also follows the $tanh$ function and that its activated output is for inputs below 0.9 identical to 0, but that for inputs higher 0.9 the activated output quickly rises to 1. Note that to obtain the binary class model prediction the final output is transformed to 1 for outputs higher than 0.5 and transformed to 0 otherwise (with a threshold of 0.5). All these findings are in line with research and state that loading times are relevant for web QoE only and that these loading times should be kept low to obtain good QoE.

5) Lessons Learned: KAN seems to be a promising solution to explainable, data-driven QoE modeling as it is trained



(a) Video Streaming QoE. (b) Web QoE.

Fig. 14: Kolmogorov-Arnold-Network.

quickly and can be interpreted easily. Further benefits of KAN are the automated optional mapping of activation functions to symbolic functions, which facilitates mathematical interpretation and bridges the gap between SR and neural networks, and the sparsity regularization and pruning techniques to reduce equation complexity. This allows KAN to identify meaningful relationships in the data, while providing rules similar to SR. In our experiments, we observed that KAN reasonably captured the impacts of the average bitrate and the total stalling duration for the video streaming QoE and the impacts of BI_{PLT} for web QoE, consistent with other models and QoE research. However, we also found that most of the resulting equations were extremely difficult and challenging to interpret, particularly with increasing network depth. An additional drawback of KAN is the huge parameter space which requires careful tuning. Overcoming these challenges is fundamental for achieving a good trade-off between model performance and interpretability with KAN. Nevertheless, KANs are still in their infancy and can be expected to develop further in the future. These developments could make KAN a more robust option for QoE modeling in the future.

I. Generalized Additive Models

1) Background: A Generalized Additive Model (GAM) is a generalized linear model in which the model output is computed by summing up each of the arbitrarily transformed input features along with a bias [122]. Thus, GAMs can be described by

$$g(\mathbb{E}[y]) = \beta + \sum_{i=1}^{n} f_i(x_i), \tag{2}$$

where x_i is the i^{th} input feature of n total features, f_i is a univariate arbitrary predictor function for feature i, y is the target variable, and g is the link function. The link function g relates the expected value of the target variable to the learned predictor functions f_1 to f_n. The form of a GAM enables a direct interpretation of the model by analyzing the learned functions f_1 to f_n and the transformed inputs, which allows to estimate the influence of a feature. In this work, we utilize two state-of-the-art ML-based GAMs to model video streaming QoE, namely, EBM [121] and NAM [122].

TABLE IX: Overview of EBM internals [139].

Step	Action
1	Select the first feature
2	Train a very small tree using the selected feature only
3	Update residuals
4	Iterate through remaining features and repeat steps 2 and 3
5	Perform multiple iterations of steps 1 to 4
6	Collect the outputs of all trees for each feature's inputs
7	Build predictor functions based on inputs and collected outputs
8	Interpret predictor functions

a) Explainable Boosting Machine: Explainable Boosting Machine (EBM) is based on DTs and gradient boosting [139], [140]. In gradient boosting, a model consists of multiple weak models, such as DTs, that iteratively improve predictions by reducing errors from previous models. The training process of EBM is shown step by step in Table IX. In each step, a small tree is trained for each feature separately, starting with the first feature and continuing in round-robin fashion. The residual (error) of the small tree for the first feature is passed to the next feature, where another small tree is trained to reduce the residual. The learning rate is so small that the order of the features has no impact. After training on all features, the next iteration begins, with a new set of small trees trained for each feature to further reduce the residual error. After n training iterations, the trees for each feature jointly provide predictions, allowing construction of the predictor functions f_i for each feature. Then, the trees are no longer needed and the predictor functions are sufficient to express the model. Due to the model's additive nature, it is easy to understand how EBM derives its output using the inputs and outputs of the predictor functions. EBM further extends GAM by also considering additional pairwise feature interaction terms ($\sum f_{i,j}(x_i, x_j)$), which are added on the right side of the GAM equation, while maintaining explainability.

b) Neural Additive Model: In contrast to EBM, the Neural Additive Model (NAM) utilizes neural networks as predictor functions f_i. The architecture of NAM is depicted in Fig. 15. The feature inputs x_1 to x_n are separately passed to

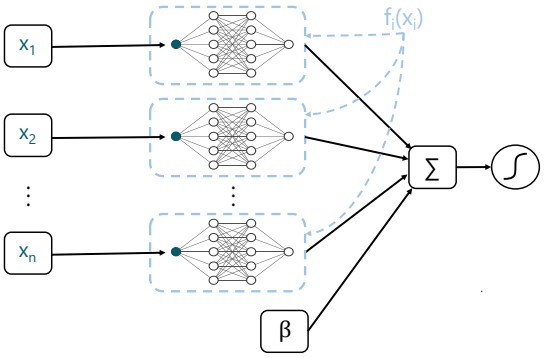


Fig. 15: Architecture of Neural Additive Model.

the corresponding neural networks, represented by f_i, i.e., each feature input has its own neural network. The neural networks each take the single feature value as input and transform it (with the help of hidden layers) to a single output. These outputs are summed together with a (learnable) bias to obtain the model output (for a regression task). If we want to perform classification, the model output has to be passed through an additional activation function, e.g., sigmoid activation function to perform binary classification. The transformation within each neural network is what the model actually learns to improve its predictions. As with EBM, the model can be easily interpreted due to its additive nature when considering the feature inputs and the outputs of each predictor functions.

2) Implementation: For our experiments, we use the EBM implementation of the Python package *interpret* from InterpretML [140]. For EBM, we test several hyperparameters during cross-validation. For NAM, we utilize the original TensorFlow code provided by the authors of NAM [141] and adapt it to our needs. We optimize the number of neurons in each layer, the number of layers, dropouts, and the learning rate during hyperparameter optimization.

3) Video Streaming: Figure 16a shows the learned predictor functions for the best set of hyperparameters for both EBM (green) and NAM (blue) on the video streaming QoE dataset. The findings are mostly in line with SHAP values, but both models provide much smoother predictor functions and are thus easier to interpret. EBM and NAM differ only marginally and mostly in areas where the data density is low. Here, EBM outperforms NAM (cf. Table X) by overfitting on single data points. We can see this, for example, for a high total stalling duration and a high number of quality switches, where at some point EBM stops the negative trend and strongly contrasts its previous trend to improve predictions for extreme outliers.

Using the smooth predictor functions, it is easy to apply curve fitting. In the bottom right plot, we fit the average bitrate predictor function of NAM, which was shifted by the average MOS of the dataset to obtain the original MOS scale on the y-axis, on an inverted x-axis using exponential (IQX), logarithmic (WQL), and linear functions (LIN). Note that this constitutes a univariate mapping of average bitrate to MOS, neglecting the other influencing factors. We observe that our predictor function follows the WQL hypothesis [70] (red) with a high $R^2 = 0.967$. This is in line with the mechanics of P.1203, where the authors of [142] showed the same logarithmic behavior for the bitrate in mode 0.

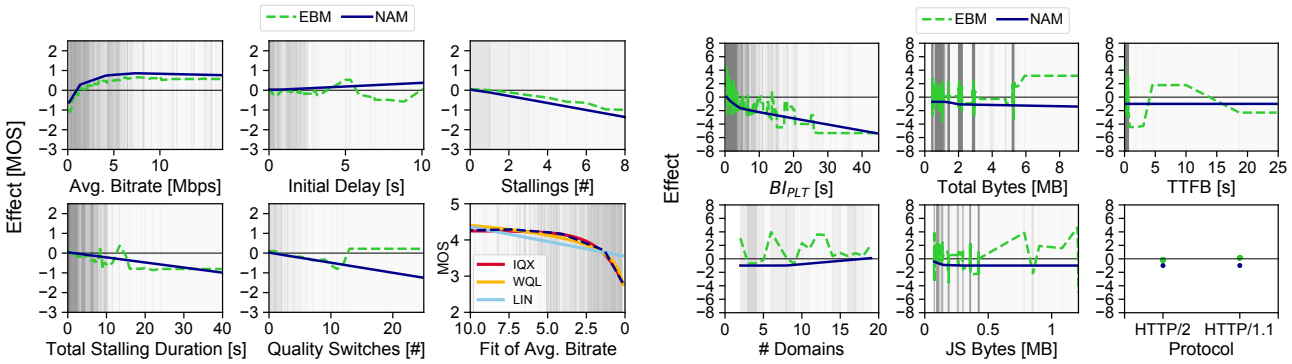
4) Web Browsing: For the web browsing use case, we show the obtained predictor functions for both NAM (blue) and EBM (green) in Fig. 16b. First of all, we observe that the predictor functions for EBM are fluctuating strongly, thereby not allowing any reasonable interpretation. Again, the model strongly overfits, thereby increasing performance, but losing interpretability.

In contrast, NAM shows again easy to interpret predictor functions. It is shown that total bytes, JS bytes, TTFB, and the HTTP protocol have a negligible mostly constant effect on the model output. In contrast, BI_{PLT} affects the model output strongly negative. This predictor function consists of two linear functions with different slopes. We observe a stronger negative slope for a BI_{PLT} range of 0 to 5 seconds, compared to the range of 5 to 40 seconds and beyond. Interestingly, NAM also attributes a small linear positive effect to the number of accessed domains. Remember that SHAP values showed the same behavior in Fig. 10b.

5) Lessons Learned: GAMs provide easy to interpret non-linear, additive functions that effectively explain fundamental relationships between QoE factors and MOS, aligning with previous data-driven models. The GAMs' univariate predictor functions allow for easy interpretability. Further, curve fitting can be applied to those functions, as demonstrated for the video streaming QoE use case, to integrate theoretical hypotheses into the modeling process. For these reasons, we consider GAMs as highly suitable for explainable, data-driven QoE modeling. Due to their ability to capture feature interactions, EBMs are more powerful than NAMs in certain scenarios. However, this power comes at the cost of overfitting and thus reduced interpretability, especially in sparse data regions. Moreover, EBM lacks the flexibility of neural networks and stochastic gradient descent as provided by NAM. These properties make NAMs better suited for context-specific modeling, managing uncertainty, and enabling collaborative learning. On the other hand, the major limitation of GAMs is the assumption that the target variable can be expressed as the sum of univariate predictor functions. If this is not the case, GAMs will not be able to capture the true underlying relationships. Summarizing, we identify NAM as most suitable solution to achieve data-driven, explainable QoE modeling.

J. Benchmarking

Next, we perform a benchmarking of the trained models for the different use cases and feature sets. In addition to the interpretable models and QoE expert models, we also consider XGBoost, RF, and MLP as baseline black-box models. Table X shows the performance of the ML and expert models per feature set for the video streaming QoE use case. For the differences between the feature sets, refer to Section V-A above. To describe the goodness of fit of the regression models, we use the Root Mean Square Error (RMSE) and the Mean



(a) Video Streaming QoE.

(b) Web QoE.

Fig. 16: Generalized Additive Models.

Absolute Error (MAE) on the MOS range, which should be as small as possible, and the coefficient of determination (R^2), which should be close to 1, but which is also unbounded to negative infinity. Note again that Hoßfeld and Mok (marked with an asterisk * in the table) do not consider adaptive video streaming.

The table shows that almost all ML-based models outperform even the best expert model P.1203 in our dataset. However, this is no surprise since the ML models are fitted directly to our dataset. Nonetheless, the expert QoE models could have been expected to generalize well over unseen data. For both feature sets, we can also see only marginal differences in both directions. This indicates that the data-driven choice of the feature set was an excellent one for our dataset. Finally, we observe that all XAI models perform worse than the black-box models, indicating that explainable models suffer a minor performance loss.

The performance of the ML-based and expert web QoE models is shown in Table XI. Here, we use accuracy, precision, recall, and F1-score as evaluation metrics as we have a binary classification task (performance close to 1 is desired).

We observe that all tree-based models (DT, RF, XGBoost) overfit the data and provide scores of 1.0 for each metric. In contrast to the strongly heterogeneous video QoE dataset, the tree-based models can easily learn the relationships between features and label, which comes as expected. In contrast, TabNet is not able to capture the relationships at all. It provides the lowest scores (F1-score: 0.29, expert features) for all considered models. Considering LogReg and NAM, we observe a similar mediocre performance in terms of evaluation scores (F1-score of 0.70). However, we also see that these models perform better than the actual expert QoE models (F1-score of 0.60). EBM outperforms NAM with a F1-score of 0.93 again strongly. This also comes as expected as EBM is better able to overfit on the data. Overall, the differences in performance between the data-driven and expert feature sets is marginal indicating again that the data-driven feature selection worked well.

TABLE X: Performance comparison of data-driven and expert video streaming QoE models for different feature sets.

	Data-Driven			Expert		
Model	RMSE	MAE	R^2	RMSE	MAE	R^2
P.1203	-	-	-	0.61	0.47	0.35
Hoßfeld*	-	-	-	1.62	1.44	-3.57
Liu	-	-	-	1.77	1.63	-4.45
Mao	-	-	-	0.73	0.61	0.35
Mok*	-	-	-	0.88	0.73	-0.35
Petrangeli	-	-	-	1.62	1.44	-3.57
XGBoost	0.25	0.15	0.89	0.23	0.15	0.90
RF	0.28	0.21	0.86	0.27	0.19	0.87
MLP	0.47	0.37	0.61	0.46	0.35	0.63
LR	0.61	0.49	0.36	0.60	0.48	0.38
DT	0.60	0.47	0.36	0.60	0.46	0.37
TabNet	0.49	0.38	0.58	0.49	0.38	0.58
SR	0.57	0.46	0.42	0.57	0.46	0.42
KAN	0.51	0.40	0.48	0.50	0.39	0.51
EBM	0.43	0.34	0.67	0.42	0.32	0.69
NAM	0.51	0.40	0.54	0.50	0.39	0.56

TABLE XI: Performance comparison of data-driven and expert web QoE models for different feature sets with respect to accuracy (A), precision (P), recall (R), and F1-score (F1).

	Data-Driven				Expert			
Model	A	P	R	F1	A	P	R	F1
IQX	-	-	-	-	0.67	0.59	0.67	0.60
WQL	-	-	-	-	0.67	0.59	0.67	0.59
XGBoost	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
RF	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
MLP	0.70	0.73	0.70	0.71	0.71	0.71	0.71	0.71
LogReg	0.71	0.71	0.71	0.71	0.70	0.70	0.70	0.70
DT	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
TabNet	0.42	0.68	0.43	0.39	0.37	0.67	0.37	0.29
SR	0.30	0.15	0.50	0.30	0.30	0.15	0.50	0.30
KAN	0.66	0.67	0.66	0.65	0.65	0.66	0.65	0.65
EBM	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93
NAM	0.71	0.72	0.71	0.26	0.68	0.71	0.68	0.69

TABLE XII: Estimated model training times.

Model	Level	Hardware	Min [s]	Max [s]
XGBoost	Fold	GPU	0.10	0.50
RF	Fold	CPU	0.01	0.60
MLP	Epoch	GPU	0.35	1.05
LinReg	Overall	CPU	3e-4	5e-4
LogReg	Overall	CPU	0.013	0.018
DT	Fold	CPU	0.003	0.008
TabNet	Epoch	GPU	1.50	4.50
SR	Round	CPU	0.05	0.07
KAN	Epoch	CPU	0.16	0.84
EBM	Fold	CPU	0.50	50.00
NAM	Epoch	GPU	0.50	1.10
SHAP	Overall	CPU	0.20	0.70
Anchors	Instance	CPU	0.08	0.12

K. Training Times

Last but not least, we shortly quantify the expected training duration of the different models in seconds in Table XII. Note that we denote the minimal and maximal training time for the models observed in our experiments. They provide a lower and upper bound for the training time, which is strongly affected by the actual values of the hyperparameters. In this table, we also denote the hardware, on which we trained the individual models. For CPU-based training, we utilize an AMD Ryzen 7 3700X 8-Core Processor, and for GPU-based training we utilize a NVIDIA GeForce RTX 2060 graphics card. Since the models differ strongly in the way they are trained, e.g., number of iterations, time-scale, or amount of data, it is not possible to compare the training times of the models perfectly. The table therefore contains a column where we state on which level we have measured training time. The training times for models without hyperparameters like LinReg and LogReg denote the overall training time, i.e., it took about 3e-4 seconds to train the model on the video streaming dataset. The training times for tree-based models like DT, RF, XGBoost, and EBM are reported per fold, i.e., how long did it take to train each model on one set of hyperparameters during hyperparameter optimization. The number of folds required, however, depends on the number and levels of hyperparameters considered during optimization. For models, which are trained on mini-batches and with stochastic gradient descent epoch-wise, i.e., MLP, TabNet, KAN, and NAM, we state the amount of time required to complete one epoch, i.e., one training round on all batched data. The overall training duration for one set of hyperparameters, however, depends on the training process if additionally other means like early stopping are used. For SHAP, the Python SHAP package computes all Tree SHAP values for the provided dataset simultaneously, i.e., it took 0.20 to 0.70s to compute all SHAP values for the video streaming QoE dataset with data-driven features. For Anchors we could only measure the time it took to generate a prediction for one instance. Note that for the post-hoc explanation techniques the training time of the model to explain has to be included in the overall training time, too.

It can be easily observed that the less complex models like LinReg, LogReg, and DT are significantly faster to train than other models. This is extremely evident when comparing the tree-based models DT, RF, XGBoost, and EBM. As RF consists of multiple DTs, it is only reasonable that the training of RF takes more training time. The results for XGBoost and EBM show that these training times rise further when additional training techniques like boosting are introduced. We explain the high training times of EBM with the fact that the training consists of multiple rounds, i.e., we basically train a RF with boosting for multiple times (in this tutorial a maximum of 5000 rounds).

When comparing the MLP with NAM, we see that the training of NAM takes a bit longer than for the MLP. This can again be explained by the model complexity, as we train in a NAM not only one MLP, but several simultaneously. With KAN, we get the lowest training times of the neural network based models. Note also that the KAN model was trained on CPU, while MLP and NAM were trained on GPU, thus actually providing an advantage to MLP and NAM. While it is possible to run KAN also on GPU, the fast training of KAN on CPU never motivated us to adopt it to GPU.

VII. Context-Aware QoE Modeling

In the previous use cases, we highlighted the suitability of additive models for obtaining easy to interpret QoE models. Such additive QoE models are also already well-established, e.g., the E-Model for speech quality assessment [143]. So far the presented NAMs have learned predictor functions which model the expected value of the underlying feature distributions only (cf. Figure 16a). In practice, however, users often perceive the same stimuli differently, leading to multiple profile or context groups [81]. While a global mean can be learned across these groups, the resulting variance often fails to accurately represent the individual contexts. To address this, separate sets of predictor functions must be learned for each context. This approach is also already applied in practice, e.g., the ITU-T expert video quality prediction models P.1203 and P.1204 [24], [144] increase MOS computation accuracy for different contexts by using context-specific coefficients for the provided contexts, e.g., video codecs (H.264, H.265, or VP9), assumed display resolution (2160p for PC/TV and 1440p for mobile/tablet), and viewing distance (1.5-3 times the screen height for PC/TV and 4-6 times the screen height for mobile/tablet). However, data-driven models generally lack support for such context-specific coefficients. Adding a context variable as input feature is insufficient as the model still learns a single set of predictor functions, shared across all contexts.

In the following analysis, we focus on NAM only. We showcase how a NAM can be extended to model QoE for different user profiles and contexts. We consider a context to be any kind of feature, which can be additionally used to separate users into different clusters and hence improve modeling accuracy overall, e.g., the user uses a mobile or

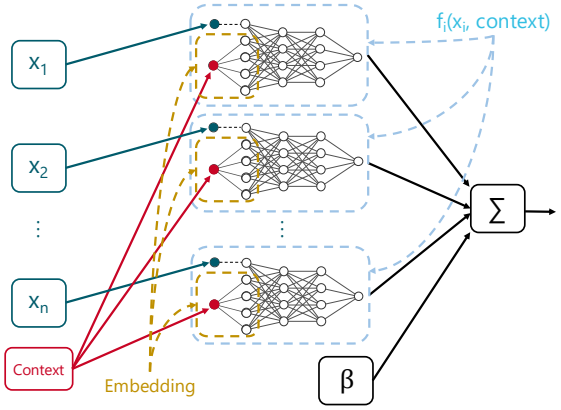


Fig. 17: NAM architecture incorporating context.

desktop device, the kind of network technology used, or the data subscription. On the other hand, we consider user profiles to be more user-related, e.g., the sensitivity of a user towards stalling events, any kind of user impairment, or demography. However, both user profiles and context can be modeled with the same approach. Next, we thus reconsider the video streaming QoE use case, but also synthesize artificial data to showcase the potential of including user profiles or context in the modeling step.

A. Contextualized Neural Additive Model

To be able to model user profiles or contexts into the Neural Additive Model (NAM), we have to adjust the architecture of NAM slightly (cf. Fig. 17). In the following, we do not distinguish anymore between user profiles and context, but only talk about context. Identical to the original NAM, we have a single subnetwork f_i for each feature x_i, where each subnetwork f_i receives one of the features as inputs. We modify the architecture such that each of the subnetworks receives the same additional input named $context$ along with the feature input x_i. This input is an arbitrary categorical variable, e.g., different user profiles with respect to sensitivity or different user contexts, e.g., mobile usage vs. desktop usage. As the context is usually a categorical variable and as it is impossible to directly compute with categorical variables in ML, we are required to convert these variables to numeric values beforehand, e.g., by simple integers. If necessary, the input $context$ is thus mapped from the categorical variable, now represented by an integer, to a low dimensional vector using an injective function $f : \mathbb{I} \rightarrow \mathbb{R}^k$. This low dimensional vector is usually called embedding. The injective function can be for example a simple one-hot encoding or it can be an embedding layer as part of a neural network. This injective function is represented by the yellow frames in the subnetworks of Fig. 17. An embedding layer is in a nutshell a lookup table with trainable weights, which is trained along with the other weights of a neural network using backpropagation. In our modified NAM architecture, we insert such an embedding layer to each subnetwork as first layer (see context embedding in the figure). The model works as follows: First, the contexts are separately mapped to the k-dimensional embeddings by each subnetwork, where k specifies the dimension of the embedding vectors (and thus the number of available parameters). These k-dimensional embeddings are then concatenated with the scalar feature input x_i of each subnetwork to the actual subnetwork inputs of dimension k+1. These subnetwork inputs are next separately transformed by the remaining layers of each subnetwork, before the outputs of the subnetworks are summed along with the bias β. The output of each subnetwork thus corresponds to the effect of a feature on the model output conditioned on a specific context.

Assuming there are multiple contexts, which we want to model, we can either scale up the architecture by adding more embedding layers within the subnetworks, each embedding layer responsible for a single context, or by simply increasing the size of the single embedding layer and increasing its capacity. If we assume k contexts, the number of inputs changes to one feature input plus k context inputs. The resulting embeddings of the contexts are then concatenated along with the feature input x_i to the new subnetwork input, resulting in an actual input of dimension $\mathbb{R}^{1+k \cdot d}$, where d is the dimensionality of the embeddings.

To obtain the interpretations for each profile, we again have to consider the feature inputs x_i along with the output of the predictor functions f_i, this time, however, also freezing the context input, i.e., passing the same context over and over until we have received the underlying predictor functions. This is repeated for each context to obtain the different context-based predictor functions.

TABLE XIII: User profiles for contextualized NAM analysis.

Profile	Description	Synthesizing Functions
A	Baseline	$\beta = 3.41$ $f1 = -1.659 \cdot \exp(-0.636 \cdot x) + 0.874$ $f2 = 0.035 \cdot x + 0.0189$ $f3 = -0.186 \cdot x + 0.008$ $f4 = -0.025 \cdot x + 0.0294$ $f5 = 18.76 \cdot \exp(-0.0026 \cdot x) - 18.73$
B	Avg. bitrate as single influence factor	$\beta = 3.41$ $f1 = -4.0 \cdot \exp(-0.50 \cdot x) + 1.5$ $f2 = 0.0$ $f3 = 0.0$ $f4 = 0.0$ $f5 = 0.0$
C	Stalling as dominant factor	$\beta = 3.41$ $f1 = 0.0$ $f2 = 0.0$ $f3 = -0.246 \cdot x + 0.2$ $f4 = -0.05 \cdot x - 0.4$ $f5 = 0.0$
D	Initial delay and quality switches as dominant factors	$\beta = 3.41$ $f1 = -0.559 \cdot \exp(-0.005 \cdot x) + 0.874$ $f2 = -0.286 \cdot x + 0.008$ $f3 = 0.0$ $f4 = 0.0$ $f5 = -0.60 \cdot \log(0.5 + x) - 0.15$

B. Dataset Generation

As our crawled video streaming QoE dataset does not contain usable profiles or contexts, we artificially augment the dataset by synthesizing user profiles. Here, we assume that users follow different profiles, where some users' QoE is only affected by the avg. bitrate or where some users' QoE is only affected by stalling-related features. This means we can reuse the samples of the original dataset, but change the label, i.e., QoE rating, by modifying the functions, which determine how the feature input is mapped to the actual effect on the model output. As we use an additive model, we only have to change the functions of the subnetworks f_i to obtain QoE ratings following desired profiles. For each profile, we simply copy the original data, compute a new label given the functions of the subnetworks, and append the data to a new dataset. This results in a dataset c times the size of the original dataset, where c corresponds to the number of distinct profiles. Keeping the same amount of data for each profile ensures here that each profile can be learned properly. If the data was not already balanced, it should have been balanced such that each context occurs equally and thus some contexts do not influence others as they contribute more during training.

As a proof of concept, we consider the four user profiles, which are listed in Table XIII. Profile A corresponds to the baseline, where we reuse the original data. We also fit the learned predictor functions from the original dataset (cf. Fig. 16a). These fitted functions for the baseline profile act as orientation point for defining the other profiles. Profile B imitates users whose subjective ratings are affected by the avg. bitrate only. Here, only the avg. bitrate subnetwork function f_1 follows an exponential function and determines the QoE rating, while the other functions f_2 to f_5 have zero impact on the model output. With profile C, the total stalling duration and the number of stalling events become the dominant QoE influence factors. Therefore, the linear subnetwork functions f_3 and f_4 strongly affect the model output, while the remaining linear functions have no effect on the model output. Finally, profile D models the initial delay and the number of quality switches as most influential QoE factors. Here, the function f_2 for the initial delay follows a linear function, while the function f_5 for the number of quality switches follows a logarithmic function. The remaining functions have again a negligible or no effect.

C. Implementation

We extend the code provided by the authors of NAM by modifying the subnetwork architecture. We add an embedding layer to the subnetworks and concatenate the output of the embedding layer and the actual feature input. The dimensionality of these embedding layers is also selected during hyperparameter optimization.

D. Video Streaming

The best performing model on this synthetic dataset achieved a R^2 score of 1.0 and a low RMSE of 0.05 and MAE of 0.03. Note that the performance is excellent here, because we used a custom dataset with known functions and without adding additional noise. We visualize the learned predictor functions of this model in Fig. 18. The learned functions are hereby colored according to their user profiles.

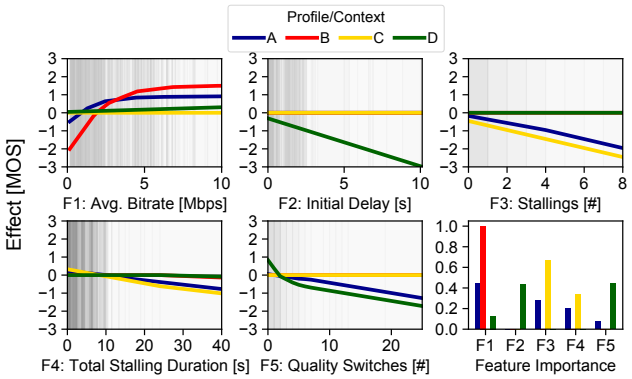


Fig. 18: Effects of expert video streaming QoE features on MOS with contextualized NAM and different user profiles.

We observe that for profile A (blue) the learned predictor functions closely resemble those in Figure 16a, indicating that the model effectively captured the original relationships despite the presence of samples from other profiles. For profile B (red), where only avg. bitrate affects the subjective rating, we observe that the predictor function for the avg. bitrate (F1) has a higher amplitude than for profile A and that it primarily shapes the model output. The stalling-related feature dominance for profile C (yellow) is easily visible in the predictor functions as mostly the number of stallings (F3) and total stalling duration (F4) affect the MOS, while the effects of average bitrate (F1) and initial delay (F2) are constant around 0. Finally, for profile D (green), we observe a strong negative linear influence of the initial delay (F2) and negative logarithmic influence of the number of quality switches (F5) as expected. The mean feature importance per profile on the bottom right confirms that the model was able to capture the underlying functions of the individual profiles.

E. Lessons Learned

In many real-world scenarios, users perceive the same stimuli differently due to varying user profiles or contexts. Integrating such profiles and contexts into the QoE model is thus fundamental to accurately model QoE. Existing QoE models, like the ITU-T recommendations P.1203 and P.1204 for video streaming, already incorporate these mechanisms. While it is theoretically possible to generate separate models for each context, such an approach lacks scalability. Therefore, our goal was to develop a single data-driven, explainable QoE model capable of learning different predictor functions for different contexts simultaneously. We demonstrated that incorporating user profiles and contexts into NAM is both straightforward and effective by encoding categorical variables numerically and integrating them through additional embedding layers within the sub-modules. In our synthetic scenario

experiments, such an adapted model successfully captured the different contextual influences on the QoE. In detail, the model accurately reproduced the original predictor functions of the different contexts, demonstrating its ability to simultaneously handle multiple context instances.

VIII. Uncertainty-Based QoE Modeling

One problem with most parametric ML models, e.g., neural networks, is that they learn a mean prediction for specific feature ranges based on the training data. However, they are not able to learn the variance of the ground truth in the training data for these specific feature ranges. This model property results directly from the loss functions used for training. If we consider for example the MSE as loss function, we can see that the residuals of the predictions are minimized, thereby enforcing that the model learns to predict the most fitting mean value over all training samples matching the feature range [145], [146]. The problem here is that the training data may include ambiguous data and in reality we would be uncertain what the real label is. This is a common problem in network planning [147] and monitoring [148], but also especially relevant for QoE modeling as users often disagree in terms of subjective scores due to different preferences. In the following, we first define the term uncertainty, before we discuss different approaches to model uncertainty in XAI-based QoE modeling. Finally, we show examples for including uncertainty in our video QoE dataset explanations.

A. Uncertainty

Uncertainty allows to grasp for which value ranges a model is confident or unsure with respect to its prediction. This provides valuable insights into the model internals and how we may support the model to improve its performance. Uncertainty is generally divided into epistemic uncertainty (model uncertainty) and aleatoric uncertainty (data uncertainty) [149]–[151].

Epistemic uncertainty corresponds to the model uncertainty. Model uncertainty usually results from a lack of training data for specific feature ranges, i.e., where the density of the data is not high enough. Examples for this problem have already been shown in Fig. 16a, where, for example, total stalling durations of 10 or more seconds have a low density. As a consequence, it is likely that the model is also uncertain here. In contrast to aleatoric uncertainty, it is thus usually possible to reduce epistemic uncertainty by collecting more training data in the relevant feature ranges.

Aleatoric uncertainty is caused by the underlying data generation process. It can be considered additional noise σ_i in the target function $y_i = f(x_i)+\sigma$, where x_i is the feature input for sample i and f is the model, i.e., the function transforming x_i to y_i. If the level of noise σ changes for different feature inputs x_i, σ is assumed to be data-dependent on sample i and has heteroscedastic aleatoric uncertainty [150]. With constant noise σ_i for all feature inputs, we have homoscedastic aleatoric uncertainty [150]. With respect to QoE data, we usually have heteroscedastic aleatoric uncertainty since different degrees of influence factor degradations also result in different score distributions [152]. Since the underlying data generation process is noisy, it is impossible to reduce or remove this kind of noise. Gathering more training data or increasing the model capacity thus does not help to improve model performance. Even though reduction of aleatoric uncertainty is impossible, quantifying it helps in understanding the training data (and its underlying data generation process) and the performance of the model.

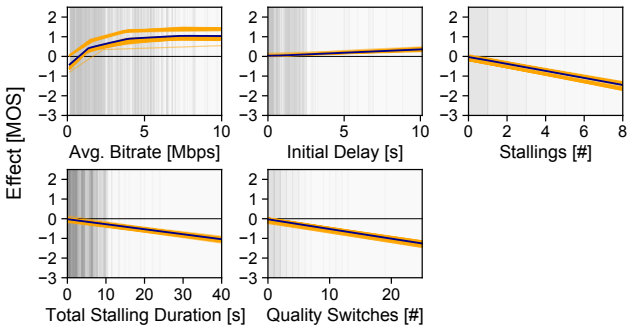


Fig. 19: Epistemic uncertainty with ensembles.

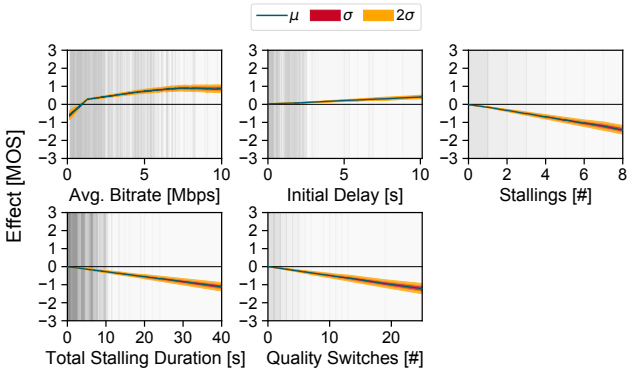


Fig. 20: Epistemic uncertainty with Monte Carlo Dropout.

There exist several approaches for quantifying (both aleatoric and epistemic) uncertainty in ML [151]. We discuss some selected and easily applicable methods to NAM in the following.

B. Ensembles

1) Background: A simple way to capture epistemic uncertainty is the use of ensembles. This means that we train a model not only once, but several times. Each time a model is trained, the weights are randomly, and thus differently, initialized, leading to slightly different models. When an ensemble model is deployed, the models of the ensemble individually return their predictions, which are then aggregated to produce the final prediction of the ensemble. In case of a regression task, the mean of all predictions is returned. If we consider a classification task, a majority vote decides the returned prediction. A prominent example for this procedure is RF, which is an ensemble of DTs. Independent of the

task, we can use the distribution of outputs to quantify the uncertainty of the model. If the predictions are all close to each other, the model is certain and epistemic uncertainty is low. If the predictions are spread far apart, the model is uncertain and epistemic uncertainty is high. As the uncertainty results from the initialized model parameters, we have epistemic uncertainty here, but cannot model aleatoric uncertainty. A benefit of this approach is that it generally works for every parameterized model.

2) Implementation: For the implementation, we simply train the original NAM from Section VI-I several times, each time however with a different random number seed. This approach was also proposed by the authors of NAM [122]. Using a different random number seed is important as these seeds result in different parameter initializations and also in different training dataset batches for each run, thus resulting in slightly different models. We train 100 NAMs for the ensemble.

3) Video Streaming: The uncertainty of the ensemble-based NAM is shown in Fig. 19. Here, an orange line is drawn for each predictor function learned for a model, resulting in 100 curves that often overlap. The blue curves state the mean predictor functions over all trained models per feature. For the predictor functions of the avg. bitrate, we can see that the predictor functions all look quite similar. The major difference between some of the runs is the offset on the y-axis. For some of the runs the lowest bitrate corresponds to an effect of 0 on the MOS, while for many other runs the effect is -0.5 or lower. As NAM is an additive model, the other predictor functions therefore also have to adjust to these offsets by adjusting their offsets. If we look at the other predictor functions, we can also see that the different models have slightly different offsets on the y-axis, but also that the predictor functions look rather identical for the different models.

4) Lessons Learned: The ensemble technique offers a general method for quantifying epistemic uncertainty in any parameterized model by initializing multiple model variants with different seeds and measuring the prediction variability of these variants. In our experiments, we observed that the ensemble technique shows how uncertain the model is with respect to its assigned effects, but that the predictor functions all look rather identical. As a consequence, this approach is not suitable to compare different models, as we can only see the uncertainty caused by the varying impacts of individual features. This means that the epistemic uncertainty modeled by ensembles is basically the uncertainty of the effect on the MOS. However, we do not see strong variations for different feature ranges what we would have actually expected.

C. Monte Carlo Dropout

1) Background: Another way to quantify epistemic uncertainty is the use of Monte Carlo Dropout [153]. Dropout is a popular technique used to increase model generalizability by randomly turning off a certain percentage of neurons in a neural network during each training step according to a Bernoulli distribution [154]. As a consequence, the model

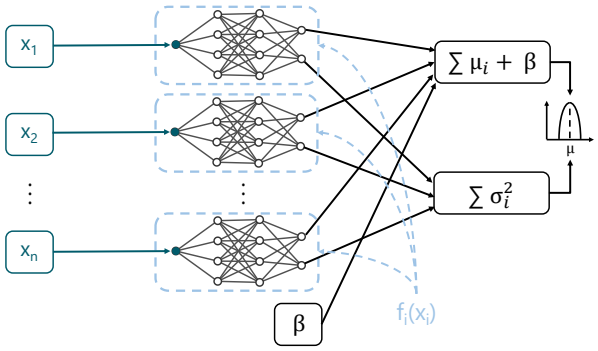


Fig. 21: NAM architecture incorporating uncertainty.

cannot rely on single neurons to learn a function, but must spread the learned information across multiple neurons and layers to become more robust against these random dropouts. Usually, dropout is only used during training and is turned off for inference. The idea behind Monte Carlo Dropout is to keep dropout active also during inference. The authors of [153] have shown that this allows to view the inference process as an approximate Bayesian inference. By keeping the dropout during inference active, we can generate multiple outputs by inferring multiple times, each time with different randomly activated dropouts. We hereby simulate the generation of multiple models, thus, roughly resembling an ensemble of models. Identical to ensembles, we can then quantify the epistemic uncertainty by considering the distribution of the outputs. Monte Carlo Dropout can be used with every kind of parametric model.

2) Implementation: For the implementation, we take again the NAM as it is, but this time keep the dropout layers active during inference. For the analysis of the Monte Carlo Dropout technique, the best hyperparameters contain a dropout rate of 10%, i.e., 10% of all neurons are switched off during a training and inference step. In total, we performed 100 predictions with the trained model to approximate mean and standard deviation of the predictor functions in a Bayesian fashion.

3) Video Streaming: The predictor functions for Monte Carlo Dropout are shown in Fig. 20. Blue lines denote the mean effects (μ) over all 100 predictions, while the red and orange areas denote uncertainty as one and two standard deviations (σ), respectively.

In contrast to the ensembles, we can see that here the uncertainty strongly depends on the x-axis, i.e., the feature values, and not on the y-axis and its offset, respectively. For all predictor functions except for the avg. bitrate, we can see that the uncertainty of the model rises with an increasing feature value. If we consider the density of the data, i.e., the grey background, we can immediately see that the model's uncertainty simply resembles the increasing sparsity of the training data. As described before, in the case of epistemic uncertainty, we can reduce the model's uncertainty at these points to improve the model. As a consequence, the model helped us in identifying value ranges, where more training

data is required. If we look at the predictor functions of the avg. bitrate, we see the same behavior for higher bitrates. Nevertheless, we also see that for lower bitrates (below 1.5 Mbps), where plenty of training data is actually available, the model is also rather uncertain. We explain this by the fact that there is much contradicting data in this range. If we consider Table III again, we can see that the LIVE Netflix I database covers only these lower avg. bitrates, but that the other metrics are similar to the other databases. As there seems to be much contradicting data, e.g., low bitrate results in many high subjective scores for the other databases, but sometimes in low subjective scores for the LIVE Netflix I database, the model is uncertain about the true effect of the feature.

4) Lessons Learned: Monte Carlo Dropout quantifies epistemic uncertainty by keeping dropout active during inference. This essentially simulates an ensemble of models to estimate prediction variability. In our experiments, Monte Carlo Dropout better quantified epistemic uncertainty for specific feature ranges in contrast to ensembles, thus proving to be superior here. In addition, it was way faster as we do not have to retrain the same model several times. However, the amount of uncertainty varies depending on the dropout rate: higher dropout rates usually result in higher uncertainties as there is automatically more uncertainty in the model present, but may subsequently also result in poor model performance. Therefore, an extensive hyperparameter optimization is required. Moreover, the technique also highlighted regions of sparse or conflicting data, offering valuable insights for refining the dataset in future iterations.

D. Probabilistic Neural Additive Model

1) Background: To quantify heteroscedastic aleatoric uncertainty, the model must estimate not only the prediction (i.e., the mean), but also the corresponding variance. In practice, this involves approximating the predictive distribution by learning its key parameters, e.g., mean and standard deviation for a Normal distribution. We extend NAM as proposed by Thielmann et al. [155] to enable the model to learn arbitrary distribution parameters, e.g., location, scale, and shape. This is achieved by adding extra output heads to each subnetwork, such that each head predicts a respective distribution parameter. Figure 21 illustrates the extended architecture using a Normal distribution. Identical to the original NAM, the effects are summed along with a bias to compute the overall mean (the original prediction). Additionally, the predicted variances from all subnetworks are summed to compute the overall variance. The overall mean and standard deviation (derived from the overall variance) then define the target distribution, from which the final prediction is sampled. Rather than using a Normal distribution, as in [155], we estimate the parameters of a Laplace distribution (location and scale) to increase robustness against noise in the training data [156]. In our experiments, we also observed that the Laplace distribution resulted in models more in line with previously learned models. As before, the predicted value is compared to the corresponding ground truth (i.e., MOS) to compute the loss during training, while the variance is implicitly learned only (i.e., without supervision). To optimally fit the distribution parameters, we minimize the negative log-likelihood of the estimated Laplace distribution. A distribution providing accurate predictions for an instance should therefore yield low variance across subnetworks. In contrast, if predictions are uncertain or inconsistent, one or more subnetworks should show high variance, reflecting the model's uncertainty due to contradicting effects from other instances. Note that this models aleatoric uncertainty only.

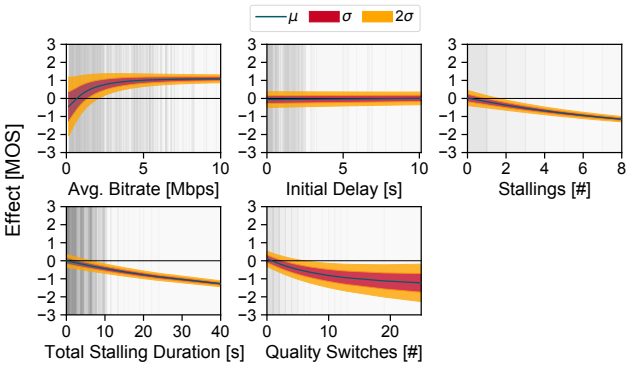


Fig. 22: Aleatoric uncertainty with probabilistic NAM.

2) Implementation: For the implementation, we extend the original NAM by adding a second head to each feature subnetwork f_i. The outputs of the subnetworks are then summed to get the location and to derive the scale parameters, respectively, and then fed into a TensorFlow Probability Laplace layer, which samples from the determined distribution during training. In contrast to the previous NAM-based models, we thus have to train this model using negative log-likelihood. Critical for an efficient learning process is also that the estimated scaled predicted by each subnetwork is higher than zero. A simple trick to avoid this is to apply the Softplus activation function on the scale head.

3) Video Streaming: We first investigate in how far the change of the loss function affects the performance. With an RMSE of 0.51, an MAE of 0.39, and a R^2 score of 0.55, we observe that the probabilistic model performs on par with the baseline NAM discussed in Table X.

Figure 22 depicts the learned predictor functions per feature along with the uncertainty learned for each data point. In line with the previous figure for Monte Carlo Dropout, we show for every feature the learned mean effect (μ, location) and the red and orange areas denote the uncertainty as one and two standard deviations (σ), respectively. First of all, the figure shows that the learned predictor functions are rather similar to the predictor functions of the baseline model. Again, the predictor function for the average bitrate (F1) follows a logarithmic relationship, the predictor functions for the number of stalling events (F3) and total stalling duration (F4) follow a negative linear trend, and the predictor function for the initial delay (F2) approximates a horizontal line close to 0. A minor exception is that the predictor function for number of quality

switches (F5) does not follow a strict linear trend any more, but seems to follow a weak exponential trend, even though the general trend is similar to the baseline.

If we look at the uncertainty indicated by the standard deviation, we see a different behavior compared to the model based on Monte Carlo Dropout. Remember, here we learned aleatoric uncertainty, in contrast to epistemic uncertainty as before. Therefore, differences between the two models are expected. For the avg. bitrate (F1), we observe that uncertainty is higher for lower bitrates and decreases for higher bitrates. If we look again at the databases in Table III, we see that, in particular, the LIVE Netflix databases operate in low bitrate ranges, while producing a MOS not completely different from the other databases. Thus, it is only logical that the probabilistic model learns that there is a lot of contradicting data, thus a higher aleatoric uncertainty. If we compare the results for this aleatoric model and for the epistemic model in Fig. 20, we observe that for the epistemic model the uncertainty is high for the data ranges, where only a few samples are available, i.e., we have model-based uncertainty, while for the aleatoric model the uncertainty is high, where a lot of data is available and the data may be contradicting, i.e., we have data-based uncertainty. A similar behavior can be observed for the number of stalling events (F3). Here, aleatoric uncertainty is slightly higher for zero and one stalling event compared to two or more stalling events. This can be explained by the fact that most samples in the dataset have zero or one stalling event, while other factors may vary and hence the MOS, resulting in an increasing uncertainty of the model here. The same applies for the total stalling duration (F4). For the number of quality switches (F5), we can see that aleatoric uncertainty strongly increases for a higher number of quality switches. If we look directly into the data, we can see that there are two samples in the dataset with 13 quality switches, no stalling events, and an avg. bitrate of 1.2 and 1.3 Mbps, respectively, but that the subjective scores are 3.25 and 4.46, so strongly diverging. For 12 quality switches, we also have samples with subjective scores of 1.29 and 3.50, so again strongly diverging. The model, thus, correctly identifies high uncertainty for these feature ranges.

4) Lessons Learned: Aleatoric uncertainty can be quantified by learning location and scale parameters of distributions, thus providing the ability to model uncertainty caused by inherent data noise or contradictions. To achieve this, we adopt an extension of NAM [155], which provides both location and scale of a Laplace distribution as output and is trained by approximating the target distribution rather than focusing on mean values only. In our experiments, the probabilistic NAM effectively captured regions of high aleatoric uncertainty, particularly in feature ranges with sufficient data. This suggests that the model recognized inherent noise in those regions, indicating that additional training samples do not help to reduce this uncertainty. We also observed that aleatoric and epistemic uncertainty highlight different things. Therefore, both should be considered to be able to take informed decisions, whether for collecting more data samples or for business decisions taking model predictions into account.

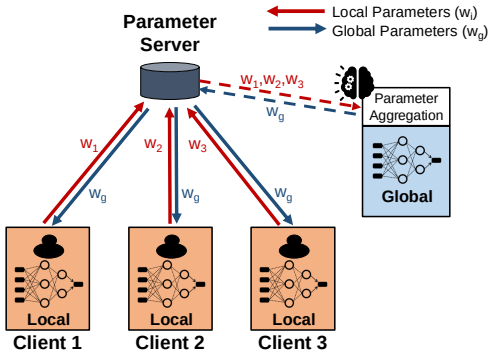


Fig. 23: Federated Learning.

IX. Data Decentralization

For successful QoE modeling, tons of data accurately describing the relationship between QoE influence factors and QoE of a service are required. However, obtaining such data is often difficult for a plethora of reasons. Further, to create a QoE model from those sensitive data, they first have to be transmitted from end devices over the network and stored in a centralized database, before a model can be created. This increases resource consumption with respect to the network and storage and also poses privacy concerns. This also indicates that QoE modeling in a centralized fashion may not always be the best solution, or may not be a feasible solution at all.

While in earlier days most ML applications worked in a centralized fashion, i.e., a model is trained on a single node with all data, ML applications have started to become decentralized, i.e., data and/or model are distributed over and trained on multiple nodes, increasing privacy preservation. When utilizing data-driven QoE modeling, this distributed aspect of modeling can also be exploited. The best known decentralized learning paradigm is Google's FL [26], which we describe next.

A. Background

With FL, each client keeps its own data locally, trains a model on its own data, and exchanges the learned model parameters or obtained gradients during training with a centralized, global parameter server only. The centralized parameter server aggregates the parameters sent by each client using a specific aggregation strategy, e.g., simply averaging the parameters, which updates the global model. This training process is also illustrated in Fig. 23. Federated training is done in multiple rounds until the global model has converged. This means that no data, except for the model parameters or gradients, have to be exchanged over a network; thus no data storage is required and privacy is also preserved for end users. This learning paradigm fits data-driven QoE modeling perfectly, as one could directly use sensitive field data without breaching privacy, while learning from the data of all end users. We also assume that additive models are highly suitable

for FL, because aggregating the subnetworks f_i for example by simple averaging should result in valuable approximations of the true global/centralized subnetworks.

By now, there exists a variety of different aggregation strategies to improve model performance in FL [157]. In the following, we discuss the selected strategies Federated Averaging (FedAvg), Federated Stochastic Gradient Descent (FedSGD) [26], and Federated Optimization in Heterogeneous Networks (FedProx) [158], which we also evaluate on our datasets. These strategies are also all implemented in TensorFlow Federated [159], which we use for all evaluations. Note that except for FedSGD all these aggregation strategies can be used in a weighted and unweighted fashion. Weighted means that the amount of available training data at a client is considered when performing the client updates on the global model, i.e., the changes are weighted for each client separately. In the unweighted fashion, all client updates are treated equally.

1) FedSGD: Federated Stochastic Gradient Descent (FedSGD) is one of the two original aggregation strategies proposed in [26]. With FedSGD, each client computes the average gradient based on the client data and the current model and passes these gradients to the parameter server. The parameter server then aggregates these gradients and performs a gradient descent step. This is identical to each client performing a gradient descent step locally and then averaging the resulting model parameters for the global model.

2) FedAvg: Federated Averaging (FedAvg) is the other aggregation strategy proposed in [26] and is a generalization of FedSGD. While FedSGD performs a single local training step only, FedAvg allows to perform multiple local training steps using a given client learning rate before updating the global model. This also means that FedAvg with a single local training step per round corresponds to FedSGD. Again, model parameters are simply averaged for the global model update.

3) FedProx: The strategy Federated Optimization in Heterogeneous Networks (FedProx) tackles the problem of heterogeneity, i.e., when the training data is non-identically distributed across the clients [158]. This is achieved by adding a proximal term to the loss function, i.e., an additional regularization term, which the parameter server can use to account for the heterogeneity caused by the partial observability. Besides this additional term, FedProx works identically to FedAvg and can be viewed as a generalization of FedAvg.

B. Implementation

For our evaluations, we use our original custom video QoE dataset. We split the custom dataset into the original databases, i.e., we simulate five different clients. We train one global NAM for the aggregation strategies FedAvg and FedProx in an unweighted and weighted fashion and a single NAM for FedSGD, i.e. five different federated NAMs in total. To have a baseline comparison for the federated NAMs, we also train federated MLPs. During hyperparameter optimization, we again tune the number of neurons per layer, the number of layers, and the dropouts for NAM and MLP. Additionally, we tune the client (local model) and server (global model) learning rates.

TABLE XIV: Performance comparison of expert and data-driven federated QoE models.

		Data-Driven			Expert		
Model	Weighted	RMSE	MAE	R^2	RMSE	MAE	R^2
MLP	-	0.47	0.37	0.61	0.46	0.35	0.63
NAM	-	0.51	0.40	0.54	0.50	0.39	0.56
MLP+FedSGD	-	0.61	0.47	0.34	0.56	0.44	0.45
MLP+FedAvg	✓	0.58	0.46	0.41	0.58	0.46	0.42
MLP+FedAvg	✗	0.61	0.48	0.36	0.65	0.51	0.27
MLP+FedProx	✓	0.73	0.58	0.06	0.58	0.45	0.41
MLP+FedProx	✗	0.63	0.52	0.32	0.67	0.53	0.21
NAM+FedSGD	-	0.61	0.50	0.34	0.56	0.45	0.45
NAM+FedAvg	✓	0.59	0.47	0.38	0.56	0.44	0.46
NAM+FedAvg	✗	0.64	0.52	0.28	0.76	0.61	-0.01
NAM+FedProx	✓	0.55	0.44	0.47	0.72	0.57	0.08
NAM+FedProx	✗	0.71	0.56	0.12	0.76	0.61	0.01

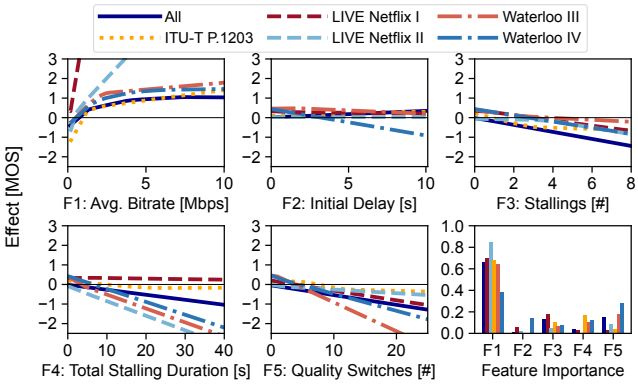


Fig. 24: Effects of expert features on MOS for the entire dataset and all databases composing the dataset with NAM.

C. Video Streaming

The results for the video streaming use case are shown in Table XIV. Note that the first two lines (models MLP and NAM) correspond to the centralized models from Table X.

First of all, we can observe that all federated models for both feature sets perform worse than their centralized versions. This indicates that the heterogeneity of the databases again causes problems for the training process.

We also see that the expert features perform slightly better than the data-driven features for both federated NAM and MLP. It is thus likely that the feature selection approach of Section V, performed in a centralized fashion, is not appropriate for the federated setting. This may be caused by the fact that the relationships between MOS and influence factors differ for each of the database as depicted in Fig. 24. As we have observed, a data-driven model provides the benefit that it is able to learn the mean response across all these databases. In case, it is important to keep the databases separate, note that we can use the databases as contexts as shown in Section VII to learn a separate model for each database.

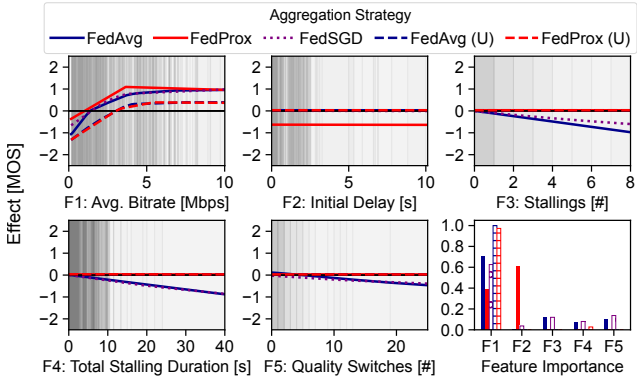


Fig. 25: Predictor functions for federated learning strategies. Unweighted strategies are shown with dashed lines.

With respect to the aggregation strategies, we can see that FedSGD and weighted FedAvg show the best results, while FedProx offers a poor performance.

Finally, we consider the explanations produced by federated NAM. Figure 25 depicts the learned predictor functions per aggregation strategy. Again, the gray shades in the background indicate the density of the data. We can see that weighted FedAvg and FedSGD conform to the shaping functions produced by the centralized NAM (cf. Figure 16a), indicating that we do not lose explainability in the federated learning setting. We can even see that the federated learning setting generates a more reasonable predictor function for the initial delay here. The centralized model considered higher initial delays to have a positive effect on the MOS, while the federated version considers the initial delay to have no influence on MOS at all. Given our domain knowledge, we know that longer initial delays may have a (low) negative impact, but never a positive effect. The federated setting thus seemed to increase the model's generalizability, but this also explains why the federated model performs slightly worse than the centralized version. The other aggregation strategies do not conform to the explanations of the centralized NAM, as they all assign low or no importance to stalling events and quality switches. This is confirmed by the feature importance plot in the lower right of the figure, showing that average bitrate has the highest impact on the output for most models, with the exception of FedProx. Additionally, only weighted FedProx considered the initial delay as impacting the outcome.

D. Practical Challenges

A variety of practical challenges arises when deploying FL for QoE modeling in reality. These challenges have already been observed in other domains such as mobile edge networks [160] and IoT [161], and are equally applicable to QoE modeling, as it represents just another use case of FL. The first challenge lies in overcoming resource constraints of end-devices. Most end-devices, such as smartphones, lack computational power and storage capacities to train complex models or to process large datasets. To prevent such issues, lightweight models like NAM are ideal as they require less resources. Additionally, techniques like random down-sampling can reduce the size of training datasets and subsequently storage requirements during training. Another problem is the battery drain caused by training, potentially leading to problems when an end-device unexpectedly shuts down during training.

A key challenge in FL settings is the heterogeneity of data across end-devices, as the data is usually not independently and identically distributed (non-iid). This means the data across end-devices does not exhibit similar statistical properties. Non-iid data usually results in strong model biases on some devices, as specific features are overrepresented, and thus leads to strong data imbalances in general. These problems complicate model training, introduce biases, and lead to slow model convergence. Mitigation strategies include data sampling techniques, custom aggregation strategies resilient to such variability, or incorporating regularization techniques to address these imbalances.

Another challenge is the efficient handling of communication overheads and synchronization of model updates caused by FL. Bandwidth limitations, high latencies, and unreliable network conditions can delay the delivery of model updates and disrupt synchronization among end-devices. Additional excessive communication overheads can strain the end-user's data plan. Asynchronous model updates can mitigate the negative impacts of the network at the cost of slower model convergence. Moreover, techniques such as quantizing, pruning, and sparsification of model updates help to reduce the network burden, improving overall efficiency.

System reliability poses another challenge. Participants may drop out unexpectedly due to battery depletion, connectivity issues, or user intervention, affecting training efficiency. Encouraging end-users to reliably participate in the federated learning process is thus fundamental and can be achieved through incentives such as financial rewards, access to exclusive services, or gamification strategies to engage users.

Finally, while federated learning reduces privacy risks, it introduces new security and privacy challenges that must be addressed during deployment. Although datasets remain local and are never shared, a malicious attacker might exploit model updates to infer sensitive information or inject malicious updates to manipulate training outcomes. Implementing more secure aggregation strategies can mitigate such risks. Even though FL can be considered a decentralized approach as all data remains distributed on the end-devices, the centralized server for aggregating participants' model updates and redistributing the global model can be considered a single point of failure, vulnerable to denial-of-service attacks that could disrupt the entire system. Decentralizing the parameter server addresses this problem, but introduces new challenges [162].

E. Lessons Learned

Successful data-driven, explainable QoE modeling requires large amounts of sensitive data. However, these data are often difficult to obtain, as they may pose privacy concerns to end-users when stored centrally. Decentralized approaches like

FL address these issues and its applicability to data-driven QoE modeling is thus of paramount interest. Our experiments showed that federated XAI-based QoE modeling is feasible using NAM and a simple aggregation strategy like weighted FedAvg, demonstrating promising results. However, efficient feature engineering poses an additional challenge as the entire dataset is not known to a central entity. Subsequently, there is also a lot of potential in optimizing the feature engineering process. Our results also indicated that there is a performance loss due to federation, but that no significant differences between black-box MLPs and white-box NAMs are to be expected. More importantly, the predictor functions of the federated NAM approximated the predictor functions of the centralized NAM well, thus maintaining model explainability even when suffering some performance losses. Summarizing, our results indicate that with careful feature engineering, NAM, and a simple aggregation strategy like FedAvg, federated XAI-based QoE models can be developed offering clear interpretability while providing competitive performance. Lastly, practical deployment of federated XAI-based QoE modeling introduces additional challenges such as limited computational resources and battery life of end-devices, non-iid data, unreliable networks, system reliability issues, and new security risks that need to be addressed before successful deployment.

X. Future Directions

A. Improvement of Explanations and Large Language Models

An open problem, on which the XAI community researches, is the general improvement of explanations produced by XAI techniques and models such that they are easily understandable and meaningful for humans. Promising research directions are here concept-based explanations [163], causal explanations in the form of counterfactual explanations [164], but also interactive and multimodal explanations.

Concept-based explanations are supposed to be easily understandable for humans as the explanations are oriented towards concepts well-known to humans. An example for such a technique is shown in [163], in which they design a technique that shows humans in an image classification task, which objects or parts of the image (concepts) the model used to generate its prediction, e.g., a wheel and the letters *police* result in the label *police car*. Identifying similar concepts in the QoE domain is, however, not trivial.

In contrast, counterfactual explanations investigate the causal relations by asking what would have happened if something else had (not) happened. Analyzing counterfactuals can thus help the user to understand how the model reasoned to come to its decision and for which scenarios it would have acted differently. A simple example would be a video streaming scenario where we want to know what would have happened to the user's QoE if there had been no stalling event during the stream.

Finally, interactive and multimodal explanations make use of the interaction with tools from different domains. One trending example is the use of Large Language Models (LLMs) in form of chatbots, allowing the user to interact with the chatbot so that the chatbot helps the user to understand the underlying model behavior [165]. There already exist several explanation dialogue systems like TalkToModel [166], ConvXAI [167], and InterroLang [168]. In [169], we took a first step towards utilizing LLMs to mediate some of the explainable models presented in this work to the end-user by proposing *QoEXplainer*.

In the future, it can be expected that these chatbots not only return text, but also for example audio-visual cues to convey their explanations.

B. Time-series QoE Modeling

While we focused on tabular data in this work by aggregating the collected service experience into session-based features, it is also possible to represent the data as time series, e.g., monitoring video quality and player state (playing or stalling) every second during a video stream [92], or incrementally tracking page load progress at fixed intervals [35]. These time series can then be input to ML or XAI models. Time-series data often contain more information on the user's experience and may thus be beneficial to better approximate QoE. One potential disadvantage of time-series based QoE modeling is that monitoring time-series data often introduces additional complexity. Also, considering the current state-of-the-art for XAI on time-series data, we have to remark that XAI techniques and models are here not yet as advanced as existing approaches for text, computer vision, and tabular data. Nevertheless, future advances in research on XAI for time-series data are going to enable explainable, time-series QoE modeling at one point.

C. Data-centric QoE Modeling

Last but not least, we want to highlight the importance of the training data. In this tutorial, we have considered XAI from the perspective of explaining the internals of a trained model only. We thus have not tried to understand how the training data affected the training process and thus the obtained model, i.e., the training data influence [170]. Attributing training data influence helps to understand which samples were actually important for the training process and which samples were considered noisy or redundant. This is particularly relevant for the QoE domain as it is often difficult to collect subjective data covering the entire range of the required feature value domains in the wild (or in laboratory studies), for example caused by rare events. Therefore, we would be interested in knowing which kind of samples should be collected to improve the model and also cover rare event edge cases effectively. The naive approach of identifying data influence is by retraining the same model several times, omitting one sample from the dataset in each round, and comparing the obtained model performance. It can be easily observed that this approach does not scale well for costly to train models and large datasets [44]. To speed this up, Koh et al. [171] adapted influence functions, a concept from statistics [172], to the machine learning domain. The influence functions by Koh

et al. simulate the influence of leaving one sample out by using the observed gradients computed during training. Influence functions and other training data attribution methods are an active research field [170]. Another field which considers the importance of the data is active learning [173]. Here, the model determines incrementally which samples it selects to train on. This selection is done by maximizing a metric, which returns the gain that can be expected from adding the sample to the dataset. Generally, fewer labelled data are also required as the active learner is allowed to ask an oracle, e.g., a human annotator, to assign a label to unlabelled data, thereby, adding the human into the training loop. Active learning is supposed to improve model performance, while reducing training time as fewer samples are required. In the past, active learning for crowdsourced QoE modeling has already been proposed [82]. No explanations for the actions of the model, however, could be derived there. An expected future direction can be the combination of data influence methods, active learning, and XAI models to thoroughly understand the entire training process and resulting model. The use of influence functions as sample selection criteria for active learning is already a relevant research topic [174].

XI. Conclusion and Lessons Learned

The importance of accurate Quality of Experience (QoE) modeling remains highly relevant to this day, in particular when observing the trend of future networks like 6G towards human-centricity [175]. However, manual QoE modeling, which requires to conduct cumbersome and non-exhaustive laboratory studies, impedes the QoE modeling process. Thus, we consider data-driven QoE modeling based on Explainable Artificial Intelligence (XAI) as a promising alternative, as it not only speeds up the entire modeling process significantly, but also mitigates the bias of researchers, and simultaneously results in understandable models due to the white-box nature of XAI models.

In this tutorial, we provided the fundamentals for research on data-driven QoE modeling with XAI. Therefore, we introduced and explained relevant concepts from QoE and XAI such that QoE researchers can build upon the collected information in the future. We especially discussed the role of explainability, context, uncertainty, and federated learning within data-driven QoE modeling and showcased different solutions to these problems.

A. Lessons Learned

We summarize the learned lessons in the following and provide a general guideline for selecting a suitable XAI technique in Fig. 26. We highlight the models, which allow to account for context adaptability and uncertainty, which are by chance the same models in this work, and which can be used for federated learning. We also denote the type of obtained explanation and for which learning tasks the respective techniques can be used.

Main effects and linear models are usually not appropriate for data-driven QoE modeling as they often assume linear relationships and subsequently miss potential underlying non-linear relationships. As a consequence, more powerful non-linear models like Decision Trees (DTs) are required. While DTs are easier to interpret than other models, they also offer a poorer performance when it gets more difficult to identify the underlying relationships, and they lose interpretability the deeper the tree becomes. Utilizing neural networks, e.g., Multilayer Perceptrons (MLPs), or an ensemble of DTs, e.g., Random Forest (RF) or eXtreme Gradient Boosting (XGBoost), are easy ways to obtain more powerful models. The resulting models, however, are black-box models and thus not interpretable anymore. To obtain insights into such a black-box model, post-hoc explanation techniques like Local Interpretable Model-Agnostic Explanations (LIME), Anchors, or SHapley Additive exPlanations (SHAP) can be used. Using such techniques is reasonable, when a black-box QoE model already exists. However, the explanations provided by the different techniques can become confusing as they often operate on perturbed feature spaces and only approximate the true black-box function. Even if the explanation technique perfectly approximates the black-box predictions, it does not guarantee that the same features are used, leading to potentially unfaithful explanations. As a consequence, we argue that inherently interpretable QoE models are more favorable when no black-box QoE model exists. This aligns with prior research [41] emphasizing the superiority of interpretable models over post-hoc explanation techniques. On the example of TabNet, we observed that more complex neural models can provide reasonable interpretations in the form of instance-based heatmaps, which come at the cost of training time. Similarly, evolutionary algorithms like Symbolic Regression (SR) are useful when exact mathematical equations are required to model the relationship between influence factors and Mean Opinion Score (MOS). Here, the resulting equations were often difficult to interpret or lacked performance and the training is time-intensive. In the end, we observed that interpretable, additive models like Kolmogorov-Arnold Network (KAN), Explainable Boosting Machine (EBM), and Neural Additive Model (NAM) indicated superior performance to all other techniques when it comes to providing insights into the learned relationships. While the recently proposed KAN is still in its infancy and the tree-based EBM is restricted in terms of modularity, NAM combined all our desired properties at the slight cost of performance (compared to the black-box models). By extending NAM we could easily integrate context adaptability and uncertainty awareness, and train the model in a federated, i.e., decentralized fashion, without losing interpretability. Note that the aspects of context adaptability, uncertainty awareness, and federated learning can be realized with most other models too, but that development complexity and costs often increase for them, e.g., realizing federated learning with tree-based models requires the implementation of custom algorithms like Federated Forest [176]. So, while such possibilities exist, these techniques are often not supported out of the box by common libraries like scikit-learn, TensorFlow/Keras, or PyTorch, and thus, require custom implementations.

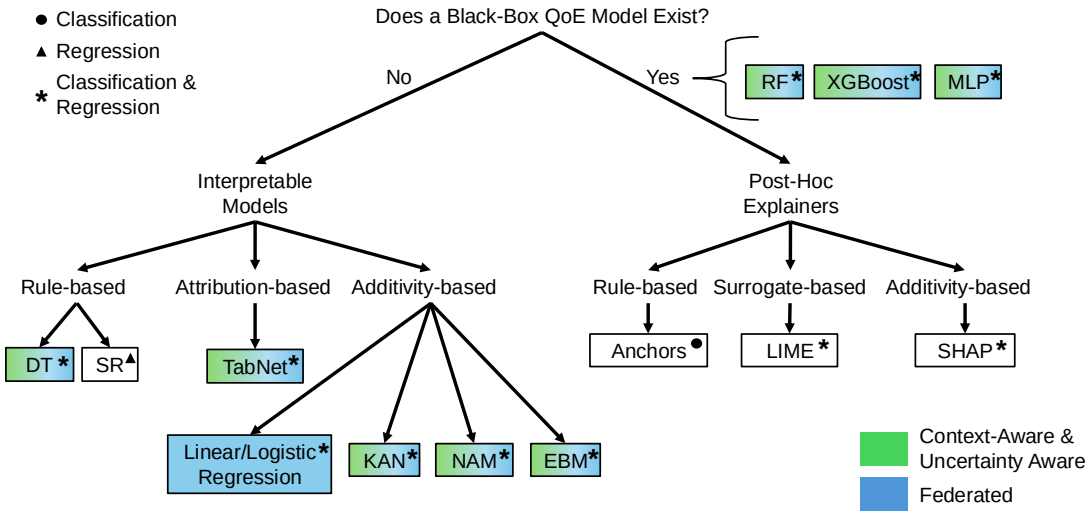


Fig. 26: Guideline on model selection.

Summarizing, we have observed that a plethora of XAI techniques and models exists to tackle all these challenges, and that all discussed techniques offered advantages and disadvantages. In our experiments, we observed that the trade-off between performance and explainability may vary depending on the considered use case. As a result, it is important to decide what degree of performance and explainability is required. If high performance is required, powerful models, such as ensemble models or neural networks, should be deployed alongside a post-hoc explainer like SHAP to provide insights. Instead, if high explainability is required, inherently interpretable models like NAM, EBM, or KAN should be deployed as they significantly ease understanding of the model internals with a slight loss of performance only. Nevertheless, Rudin [41] even argues that there is not necessarily a trade-off between performance and interpretability. With well-structured data and meaningful features, the performance gap between complex and and simpler models vanishes, while interpretability improves. Consequently, the extent of the trade-off depends on the use case and, in the broadest sense, on the domain knowledge.

B. Concluding Remarks

QoE modeling is essential for understanding and improving user satisfaction with network and service performance. By accurately predicting QoE, providers can proactively enhance service quality, optimize network resource management, and reduce customer churn. However, traditional black-box models often lack transparency, making it difficult to interpret and trust their decisions.

In this context, XAI may assume a crucial role. By incorporating XAI into QoE modeling, network and service providers can gain actionable insights into the factors affecting user experience. XAI enables better decision-making by offering clear explanations for QoE predictions, improving trust, accountability, and compliance with industry regulations.

Our extensive evaluation on the use cases of video streaming and web QoE revealed that explainable data-driven QoE modeling is ready for deployment for both classification and regression tasks. While we discussed only video streaming and web QoE, we want to clarify that all presented approaches are easily transferable to other domains, e.g., speech quality, gaming, or Virtual Reality (VR). Further, we outlined future directions and underlined that there are plenty of topics to research in the future with respect to data-driven QoE modeling.

To accelerate the adoption of explainable, data-driven QoE modeling in the future, we also provide Python Jupyter Notebooks containing all XAI techniques described in this tutorial. These Jupyter Notebooks represent a significant contribution by providing standalone, executable scripts with extensive documentation. Each notebook discusses a specific technique introduced throughout this work (cf. Fig. 2), allowing other researchers to perform rapid prototyping for custom datasets and different use cases.

ACKNOWLEDGEMENT

This work was partly funded by Deutsche Forschungsgemeinschaft (DFG) under grant SE 3163/3-1, project number: 500105691. The authors alone are responsible for the content.

REFERENCES

[1] S. Dang, O. Amin, B. Shihada, and M.-S. Alouini, "What Should 6G Be?" *Nature Electronics*, vol. 3, no. 1, pp. 20–29, 2020.
[2] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The Roadmap to 6G: AI Empowered Wireless Networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.
[3] H. Afifi, S. Pochaba, A. Boltres, D. Laniewski, J. Haberer, P. Leonard, R. Poorzare, D. Stolpmann, N. Wehner, A. Redder, E. Samikwa, and M. Seufert, "Machine Learning with Computer Networks: Techniques, Datasets and Models," *IEEE Access*, 2024.

[4] K. Brunnström, S. A. Beker, K. De Moor, A. Dooms, S. Egger, M.-N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, M.-C. Larabi *et al.*, "Qualinet White Paper on Definitions of Quality of Experience," 2013.

[5] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of YouTube QoE via Crowdsourcing," in *2011 IEEE International Symposium on Multimedia*. IEEE, 2011, pp. 494–499.

[6] J. De Vriendt, D. De Vleeschauwer, and D. Robinson, "Model for Estimating QoE of Video Delivered Using HTTP Adaptive Streaming," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, 2013, pp. 1288–1293.

[7] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a Predictive Model of Quality of Experience for Internet Video," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 339–350, 2013.

[8] Y. Chen, K. Wu, and Q. Zhang, "From QoS to QoE: A Tutorial on Video Quality Assessment," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 1126–1165, 2014.

[9] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Communications Surveys & Tutorials*, 2015.

[10] W. Song and D. W. Tjondronegoro, "Acceptability-Based QoE Models for Mobile Video," *IEEE Transactions on Multimedia*, vol. 16, no. 3, pp. 738–750, 2014.

[11] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of Quality of Experience of Video-on-Demand Services: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2015.

[12] T. Zhao, Q. Liu, and C. W. Chen, "QoE in Video Transmission: A User Experience-Driven Strategy," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 285–302, 2016.

[13] N. Barman and M. G. Martini, "QoE Modeling for HTTP Adaptive Video Streaming – A Survey and Open Challenges," *IEEE Access*, 2019.

[14] T. Hoßfeld, S. Biedermann, R. Schatz, A. Platzer, S. Egger, and M. Fiedler, "The Memory Effect and its Implications on Web QoE Modelling," in *2011 23rd International Teletraffic Congress (ITC)*. IEEE, 2011, pp. 103–110.

[15] S. Egger, P. Reichl, T. Hoßfeld, and R. Schatz, ""Time is Bandwidth"? Narrowing the Gap between Subjective Time Perception and Quality of Experience," in *2012 IEEE International Conference on Communications (ICC)*, 2012, pp. 1325–1330.

[16] A. Balachandran, V. Aggarwal, E. Halepovic, J. Pang, S. Seshan, S. Venkataraman, and H. Yan, "Modeling Web Quality-of-Experience on Cellular Networks," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, 2014, pp. 213–224.

[17] A. Sackl, P. Casas, R. Schatz, L. Janowski, and R. Irmer, "Quantifying the Impact of Network Bandwidth Fluctuations and Outages on Web QoE," in *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 2015, pp. 1–6.

[18] S. Baraković and L. Skorin-Kapov, "Survey of Research on Quality of Experience Modelling for Web Browsing," *Quality and User Experience*, vol. 2, pp. 1–31, 2017.

[19] T. Hoßfeld, F. Metzger, and D. Rossi, "Speed Index: Relating the Industrial Standard for User Perceived Web Performance to Web QoE," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2018, pp. 1–6.

[20] M. Lycett and O. Radwan, "Developing a Quality of Experience (QoE) Model for Web Applications," *Information Systems Journal*, vol. 29, no. 1, pp. 175–199, 2019.

[21] N. Wehner, M. Amir, M. Seufert, R. Schatz, and T. Hoßfeld, "A Vital Improvement? Relating Google's Core Web Vitals to Actual Web QoE," in *2022 14th International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2022, pp. 1–6.

[22] N. Wehner, M. Seufert, R. Schatz, and T. Hoßfeld, "Do You Agree? Contrasting Google's Core Web Vitals and the Impact of Cookie Consent Banners with Actual Web QoE," *Quality and User Experience*, vol. 8, no. 1, p. 5, 2023.

[23] N. Wehner, A. Seufert, T. Hoßfeld, and M. Seufert, "Explainable data-driven QoE modelling with XAI," in *2023 15th International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2023, pp. 7–12.

[24] W. Robitza, S. Göring, A. Raake, D. Lindegren, G. Heikkilä, J. Gustafsson, P. List, B. Feiten, U. Wüstenhagen, M.-N. Garcia *et al.*, "HTTP Adaptive Streaming QoE Estimation with ITU-T Rec. P. 1203: Open Databases and Software," in *ACM MMSys*, 2018.

[25] A. Rehman, K. Zeng, and Z. Wang, "Display Device-Adapted Video Quality-of-Experience Assessment," in *Human Vision and Electronic Imaging XX*, vol. 9394. SPIE, 2015, pp. 27–37.

[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[27] International Telecommunication Union, "ITU-T Recommendation E.800: Definitions of Terms Related to Quality of Service," 2008.

[28] P. Le Callet, S. Möller, and A. Perkis (eds), "Qualinet White Paper on Definitions of Quality of Experience," European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), Tech. Rep., 2013.

[29] T. Hoßfeld, P. E. Heegaard, M. Varela, and S. Möller, "QoE Beyond the MOS: Added Value Using Quantiles and Distributions," in *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 2015, pp. 1–6.

[30] International Telecommunication Union, "ITU-T Recommendation P.800: Methods for Subjective Determination of Transmission Quality," 1996.

[31] P. Casas, A. D'Alconzo, F. Wamser, M. Seufert, B. Gardlo, A. Schwind, P. Tran-Gia, and R. Schatz, "Predicting QoE in Cellular Networks using Machine Learning and in-Smartphone Measurements," in *9th International Conference on Quality of Multimedia Experience (QoMEX)*, 2017, pp. 1–6.

[32] P. Casas, M. Seufert, N. Wehner, A. Schwind, and F. Wamser, "Enhancing Machine Learning Based QoE Prediction by Ensemble Models," in *ICDCS 3rd Workshop on QoE-based Analysis and Management of Data Communication Networks (Internet-QoE)*. IEEE, 2018, pp. 1642–1647.

[33] T. Hoßfeld, S. Wunderer, A. Beyer, A. Hall, A. Seufert, C. Gassner, F. Guillemin, F. Wamser, K. Wascinski, M. Hirth, M. Seufert, P. Casas, P. Tran-Gia, W. Robitza, W. Wascinski, Z. B. Houidi, "White Paper on Crowdsourced Network and QoE Measurements – Definitions, Use Cases and Challenges," Würzburg, Tech. Rep. 10.25972/OPUS-20232, 3 2020.

[34] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li, "Vicrypt to the Rescue: Real-time, Machine-learning-driven Video-QoE monitoring for Encrypted Streaming Traffic," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2007–2023, 2020.

[35] N. Wehner, M. Seufert, J. Schuler, S. Wassermann, P. Casas, and T. Hossfeld, "Improving Web QoE Monitoring for Encrypted Network Traffic Through Time Series Modeling," *ACM SIGMETRICS Performance Evaluation Review*, vol. 48, no. 4, pp. 37–40, 2021.

[36] S. Schwarzmann, C. C. Marquezan, R. Trivisonno, S. Nakajima, V. Barriac, and T. Zinner, "ML-Based QoE Estimation in 5G Networks Using Different Regression Techniques," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3516–3532, 2022.

[37] M. Seufert, K. Dietz, N. Wehner, S. Geißler, J. Schüler, M. Wolz, A. Hotho, P. Casas, T. Hoßfeld, and A. Feldmann, "Marina: Realizing ML-Driven Real-Time Network Traffic Monitoring at Terabit Scale," *IEEE Transactions on Network and Service Management*, 2024.

[38] M. Seufert, S. Wassermann, and P. Casas, "Considering user behavior in the quality of experience cycle: towards proactive qoe-aware traffic management," *IEEE Communications Letters*, vol. 23, no. 7, pp. 1145–1148, 2019.

[39] A. Adadi and M. Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[40] A. Das and P. Rad, "Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey," *arXiv preprint arXiv:2006.11371*, 2020.

[41] C. Rudin, "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[42] D. Doran, "What does Explainable AI Really Mean? A New Conceptualization of Perspectives," *arXiv preprint arXiv:1710.00794*, 2017.

[43] A. B. Arrieta, N. Díaz-Rodríguez *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI," *Information Fusion*, 2020.

[44] C. Molnar, G. Casalicchio, and B. Bischl, "Interpretable Machine Learning–A Brief History, State-of-the-Art and Challenges," in *Joint*

European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2020, pp. 417–431.

[45] G. Kougioumtzidis, V. Poulkov, Z. D. Zaharis, and P. I. Lazaridis, "A Survey on Multimedia Services QoE Assessment and Machine Learning-Based Prediction," *IEEE Access*, 2022.

[46] J. Skowronek, A. Raake, G. H. Berndtsson, O. S. Rummukainen, P. Usai, S. N. Gunkel, M. Johanson, E. A. Habets, L. Malfait, D. Lindero *et al.*, "Quality of Experience in Telemeetings and Videoconferencing: A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 63 885–63 931, 2022.

[47] J. Baraković Husić, S. Baraković, E. Cero, N. Slamnik, M. Oćuz, A. Dedović, and O. Zupčić, "Quality of Experience for Unified Communications: A Survey," *International Journal of Network Management*, vol. 30, no. 3, p. e2083, 2020.

[48] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen, "A Survey of Emerging Concepts and Challenges for QoE Management of Multimedia Services," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2018.

[49] P. Casas and R. Schatz, "Quality of Experience in Cloud Services: Survey and Measurements," *Computer Networks*, vol. 68, pp. 149–165, 2014.

[50] S. Aroussi and A. Mellouk, "Survey on Machine Learning-Based QoE-QoS Correlation Models," in *2014 International Conference on Computing, Management and Telecommunications (ComManTel)*. IEEE, 2014, pp. 200–204.

[51] M. Alreshoodi and J. Woods, "Survey on QoE/QoS Correlation Models for Multimedia Services," *arXiv preprint arXiv:1306.0221*, 2013.

[52] S. Jelassi, G. Rubino, H. Melvin, H. Youssef, and G. Pujolle, "Quality of Experience of VoIP Service: A Survey of Assessment Approaches and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 491–513, 2012.

[53] A. Nascita, G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "A Survey on Explainable Artificial Intelligence for Internet Traffic Classification and Prediction, and Intrusion Detection," *IEEE Communications Surveys & Tutorials*, 2024.

[54] S. Wang, M. A. Qureshi, L. Miralles-Pechuán, T. Huynh-The, T. R. Gadekallu, and M. Liyanage, "Explainable AI for 6G Use Cases: Technical Aspects and Research Challenges," *IEEE Open Journal of the Communications Society*, 2024.

[55] B. Brik, H. Chergui, L. Zanzi, F. Devoti, A. Ksentini, M. S. Siddiqui, X. Costa-Pèrez, and C. Verikoukis, "Explainable AI in 6G O-RAN: A Tutorial and Survey on Architecture, Use Cases, Challenges, and Future Research," *IEEE Communications Surveys & Tutorials*, 2024.

[56] T. Senevirathna, V. H. La, S. Marchal, B. Siniarski, M. Liyanage, and S. Wang, "A Survey on XAI for 5G and Beyond Security: Technical Aspects, Challenges and Research Directions," *IEEE Communications Surveys & Tutorials*, 2024.

[57] N. Moustafa, N. Koroniotis, M. Keshk, A. Y. Zomaya, and Z. Tari, "Explainable Intrusion Detection for Cyber Defences in the Internet of Things: Opportunities and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1775–1807, 2023.

[58] I. Kök, F. Y. Okay, Ö. Muyanlı, and S. Özdemir, "Explainable Artificial Intelligence (XAI) for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14 764–14 779, 2023.

[59] G. Rjoub, J. Bentahar, O. A. Wahab, R. Mizouni, A. Song, R. Cohen, H. Otrok, and A. Mourad, "A Survey on Explainable Artificial Intelligence for Cybersecurity," *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 5115–5140, 2023.

[60] E. Cambria, L. Malandri, F. Mercorio, M. Mezzanzanica, and N. Nobani, "A Survey on XAI and Natural Language Explanations," *Information Processing & Management*, vol. 60, no. 1, p. 103111, 2023.

[61] D. Minh, H. X. Wang, Y. F. Li, and T. N. Nguyen, "Explainable Artificial Intelligence: A Comprehensive Review," *Artificial Intelligence Review*, pp. 1–66, 2022.

[62] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, and N. Díaz-Rodríguez, "Explainable Artificial Intelligence (XAI) on Time-Series Data: A Survey," *arXiv preprint arXiv:2104.00950*, 2021.

[63] M. Sahakyan, Z. Aung, and T. Rahwan, "Explainable Artificial Intelligence for Tabular Data: A Survey," *IEEE Access*, vol. 9, pp. 135 392–135 422, 2021.

[64] G. Vilone and L. Longo, "Explainable Artificial Intelligence: A Systematic Review," *arXiv preprint arXiv:2006.00093*, 2020.

[65] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable Artificial Intelligence: A Survey," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018, pp. 0210–0215.

[66] A. Ahmad, A. B. Mansoor, A. A. Barakabitze, A. Hines, L. Atzori, and R. Walshe, "Supervised-Learning-Based QoE Prediction of Video Streaming in Future Networks: A Tutorial with Comparative Study," *IEEE Communications Magazine*, vol. 59, no. 11, pp. 88–94, 2021.

[67] A. A. Barakabitze, N. Barman, A. Ahmad, S. Zadtootaghaj, L. Sun, M. G. Martini, and L. Atzori, "QoE Management of Multimedia Streaming Services in Future Networks: A Tutorial and Survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 526–565, 2019.

[68] A. Bennetot, I. Donadello, A. E. Qadi, M. Dragoni, T. Frossard, B. Wagner, A. Saranti, S. Tulli, M. Trocan, R. Chatila *et al.*, "A Practical Tutorial on Explainable AI Techniques," *arXiv preprint arXiv:2111.14260*, 2021.

[69] C. Mencar and J. M. Alonso, "Paving the Way to Explainable Artificial Intelligence with Fuzzy Modeling: Tutorial," in *International Workshop on Fuzzy Logic and Applications*. Springer, 2018, pp. 215–227.

[70] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, "Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience," in *Data Traffic Monitoring and Analysis*, 2013.

[71] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao, "Deriving and Validating User Experience Model for DASH Video Streaming," *IEEE Transactions on Broadcasting*, 2015.

[72] H. Mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," in *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2017.

[73] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the Quality of Experience of HTTP Video Streaming," in *IFIP/IEEE International Conference on Network and Service Management (IM)*, 2011.

[74] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck, "QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2015.

[75] D. Z. Rodríguez, R. L. Rosa, E. C. Alfaia, J. I. Abrahão, and G. Bressan, "Video Quality Metric for Streaming Service Using DASH Standard," *IEEE Transactions on Broadcasting*, 2016.

[76] A. Seufert, F. Wamser, D. Yarish, H. Macdonald, and T. Hoßfeld, "QoE Models in the Wild: Comparing Video QoE Models Using a Crowdsourced Data Set," in *IEEE QoMEX*, 2021.

[77] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 187–198.

[78] N. Wehner, T. Karagioules, E. Halepovic, F. Simonovski, T. Hossfeld, and M. Seufert, "To cap or not to cap: Bandwidth capping effects on network interactions and qoe of competing short video streams," in *Proceedings of the 16th ACM Multimedia Systems Conference*, 2025, pp. 90–100.

[79] E. Bocchi, L. De Cicco, and D. Rossi, "Measuring the Quality of Experience of Web Users," *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 4, pp. 8–13, 2016.

[80] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A Generic Quantitative Relationship between Quality of Experience and Quality of Service," *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010.

[81] F. Salutari, D. Da Hora, M. Varvello, R. Teixeira, V. Christophides, and D. Rossi, "Implications of the Multi-Modality of User Perceived Page Load Time," in *Proceedings of the 13th International Conference on Quality of Multimedia Experience*. IEEE, 2020, pp. 1–6.

[82] H.-S. Chang, C.-F. Hsu, T. Hoßfeld, and K.-T. Chen, "Active Learning for Crowdsourced QoE Modeling," *IEEE Transactions on Multimedia*, vol. 20, no. 12, pp. 3337–3352, 2018.

[83] M. Fiedler, U. Chapala, and S. Peteti, "Modeling Instantaneous Quality of Experience Using Machine Learning of Model Trees," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2019, pp. 1–6.

[84] S. Porcu, A. Floris, and L. Atzori, "CB-FL: Cluster-Based Federated Learning Applied to Quality of Experience Modeling," in *2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, 2022, pp. 585–591.

[85] ——, "A Clustered Federated Learning Approach for Estimating the Quality of Experience of Web Users," in *2023 IEEE International*

Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW). IEEE, 2023, pp. 1–5.

[86] S. Ickin, K. Vandikas, and M. Fiedler, "Privacy Preserving QoE Modeling Using Collaborative Learning," in *Proceedings of the 4th Internet-QoE Workshop on QoE-Based Analysis and Management of Data Communication Networks*, 2019, pp. 13–18.

[87] Z. Duanmu, A. Rehman, and Z. Wang, "A Quality-of-Experience Database for Adaptive Video Streaming," *IEEE Trans. Broadcast*, 2018.

[88] T. Hoßfeld, R. Schatz, and S. Egger, "SOS: The MOS is Not Enough!" in *2011 Third International Workshop on Quality of Multimedia Experience (QoMEX)*, 2011, pp. 131–136.

[89] K. Mitra, A. Zaslavsky, and C. Åhlund, "Context-Aware QoE Modelling, Measurement, and Prediction in Mobile Computing Systems," *IEEE Transactions on Mobile Computing*, vol. 14, no. 5, pp. 920–936, 2013.

[90] M. Seufert, O. Zach, T. Hoßfeld, M. Slanina, and P. Tran-Gia, "Impact of Test Condition Selection in Adaptive Crowdsourcing Studies on Subjective Quality," in *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2016, pp. 1–6.

[91] C. Gutterman, K. Guo, S. Arora, X. Wang, L. Wu, E. Katz-Bassett, and G. Zussman, "Requet: Real-Time QoE Detection for Encrypted YouTube Traffic," in *Proceedings of the 10th ACM Multimedia Systems Conference*, 2019, pp. 48–59.

[92] M. Seufert, P. Casas, N. Wehner, L. Gang, and K. Li, "Stream-Based Machine Learning for Real-Time QoE Analysis of Encrypted Video Streaming Traffic," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2019, pp. 76–81.

[93] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "A Machine Learning Approach to Classifying YouTube QoE Based on Encrypted Network Traffic," *Multimedia Tools and Applications*, vol. 76, pp. 22 267–22 301, 2017.

[94] A. Huet, A. Saverimoutou, Z. B. Houidi, H. Shi, S. Cai, J. Xu, B. Mathieu, and D. Rossi, "Revealing QoE of Web Users from Encrypted Network Traffic," in *2020 IFIP Networking Conference (Networking)*. IEEE, 2020, pp. 28–36.

[95] V. Vasilev, J. Leguay, S. Paris, L. Maggi, and M. Debbah, "Predicting QoE Factors with Machine Learning," in *IEEE ICC*, 2018.

[96] A. Renda, P. Ducange, G. Gallo, and F. Marcelloni, "XAI Models for Quality of Experience Prediction in Wireless Networks," in *2021 IEEE International Conference on Fuzzy Systems*, 2021.

[97] Z. Duanmu, W. Liu, Z. Li, D. Chen, Z. Wang, Y. Wang, and W. Gao, "Assessing the Quality-of-Experience of Adaptive Bitrate Video Streaming," *arXiv Preprint arXiv:2008.08804*, 2020.

[98] C. G. Bampis, Z. Li, A. K. Moorthy, I. Katsavounidis, A. Aaron, and A. C. Bovik, "Study of Temporal Effects on Subjective Video Quality of Experience," *IEEE TIP*, 2017.

[99] C. G. Bampis, Z. Li, I. Katsavounidis, T.-Y. Huang, C. Ekanadham, and A. C. Bovik, "Towards Perceptually Optimized End-to-End Adaptive Video Streaming," *arXiv Preprint arXiv:1808.03898*, 2018.

[100] R. Schatz and S. Egger, "An Annotated Dataset for Web Browsing QoE," in *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 2014, pp. 61–62.

[101] D. N. Da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the Gap Between QoS Metrics and Web QoE Using Above-the-Fold Metrics," in *Passive and Active Measurement: 19th International Conference, PAM 2018, Berlin, Germany, March 26–27, 2018, Proceedings 19*. Springer, 2018, pp. 31–43.

[102] D. Masters and C. Luschi, "Revisiting Small Batch Training for Deep Neural Networks," *arXiv Preprint arXiv:1804.07612*, 2018.

[103] D. T. Schroeder, K. Styp-Rekowski, F. Schmidt, A. Acker, and O. Kao, "Graph-Based Feature Selection Filter Utilizing Maximal Cliques," in *IEEE SNAMS*, 2019.

[104] M. A. Hall, "Correlation-based Feature Selection for Machine Learning," Ph.D. dissertation, The University of Waikato, 1999.

[105] K. Yan and D. Zhang, "Feature Selection and Analysis on Correlated Gas Sensor Data with Recursive Feature Elimination," *Sensors and Actuators B: Chemical*, vol. 212, pp. 353–363, 2015.

[106] A. M. Salih, I. B. Galazzo, Z. Raisi-Estabragh, S. E. Petersen, G. Menegaz, and P. Radeva, "Characterizing the Contribution of Dependent Features in XAI Methods," *IEEE Journal of Biomedical and Health Informatics*, 2024.

[107] D. E. Farrar and R. R. Glauber, "Multicollinearity in regression analysis: The problem revisited," *The Review of Economics and Statistics*, pp. 92–107, 1967.

[108] K. Dietz, M. Hajizadeh, J. Schleicher, N. Wehner, S. Geißler, P. Casas, M. Seufert, and T. Hoßfeld, "Agree to Disagree: Exploring Consensus of XAI Methods for ML-based NIDS," in *2024 20th International Conference on Network and Service Management (CNSM)*. IEEE, 2024, pp. 1–7.

[109] P. Casas, S. Wassermann, N. Wehner, M. Seufert, and T. Hossfeld, "Not All Web Pages Are Born the Same: Content-Tailored Learning for Web QoE Inference," in *2022 IEEE International Symposium on Measurements & Networking (M&N)*. IEEE, 2022, pp. 1–6.

[110] R. O. Kuehl, "Design of Experiments: Statistical Principles of Research Design and Analysis," *No Title*, 2000.

[111] T. D. Wickens and G. Keppel, *Design and Analysis: A Researcher's Handbook*. Pearson Prentice-Hall, Upper Saddle River, NJ, 2004.

[112] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*. John Wiley & Sons, 2021.

[113] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*. John Wiley & Sons, 2013.

[114] L. Breiman, *Classification and Regression Trees*. Routledge, 2017.

[115] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?" Explaining the Predictions of Any Classifier," in *ACM SIGKDD*, 2016.

[116] ——, "Anchors: High-Precision Model-Agnostic Explanations," in *AAAI*, 2018.

[117] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," *NIPS*, 2017.

[118] S. Ö. Arik and T. Pfister, "TabNet: Attentive Interpretable Tabular Learning," in *AAAI*, 2021.

[119] D. A. Augusto and H. J. Barbosa, "Symbolic Regression via Genetic Programming," in *Proceedings. Vol. 1. Sixth Brazilian Symposium on Neural Networks*. IEEE, 2000, pp. 173–178.

[120] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "Kan: Kolmogorov-Arnold Networks," *arXiv Preprint arXiv:2404.19756*, 2024.

[121] H. Nori, S. Jenkins, P. Koch, and R. Caruana, "InterpretML: A Unified Framework for Machine Learning Interpretability," *arXiv Preprint arXiv:1909.09223*, 2019.

[122] R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton, "Neural Additive Models: Interpretable Machine Learning with Neural Nets," *NIPS*, 2021.

[123] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[124] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *ACM SIGKDD*, 2016.

[125] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer Perceptron and Neural Networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, pp. 579–588, 2009.

[126] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-Generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631.

[127] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[128] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[129] ——, *C4.5: Programs for Machine Learning*. Elsevier, 2014.

[130] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning Important Features Through Propagating Activation Differences," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3145–3153.

[131] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation," *PloS One*, vol. 10, no. 7, p. e0130140, 2015.

[132] L. S. Shapley *et al.*, "A Value for N-Person Games," *Contribution to the Theory of Games*, 1953.

[133] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From Local Explanations to Global Understanding with Explainable AI for Trees," *Nature Machine Intelligence*, vol. 2, no. 1, pp. 56–67, 2020.

[134] Scott Lundberg, "Welcome to the SHAP Documentation," Accessed: July 17, 2024. [Online]. Available: https://shap.readthedocs.io/en/latest/index.html

[135] Marco Tulio Correia Ribeiro, "Anchor," Accessed: July 17, 2024. [Online]. Available: https://github.com/marcotcr/anchor

[136] Dreamquark, "TabNet: Attentive Interpretable Tabular Learning," Accessed: July 17, 2024. [Online]. Available: https://dreamquark-ai.github.io/tabnet/generated_docs/README.html#tabnet-attentive-interpretable-tabular-learning

[137] M. Cranmer, "Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl," *arXiv Preprint arXiv:2305.01582*, 2023.

[138] Ziming Liu, "Welcome to Kolmogorov Arnold Network (KAN) Documentation!" Accessed: July 17, 2024. [Online]. Available: https://kindxiaoming.github.io/pykan/

[139] Microsoft Developer, "The Science Behind InterpretML: Explainable Boosting Machine," 2021. [Online]. Available: https://www.youtube.com/watch?v=MREiHgHgl0k&t=157s

[140] InterpretML Contributors, "InterpretML," Accessed: July 17, 2024. [Online]. Available: https://interpret.ml/docs/index.html

[141] Google Research, "Neural Additive Models: Interpretable Machine Learning with Neural Nets," Accessed: July 17, 2024. [Online]. Available: https://github.com/google-research/google-research/tree/master/neural_additive_models

[142] M. Seufert, N. Wehner, and P. Casas, "Studying the Impact of HAS QoE Factors on the Standardized QoE Model P. 1203," in *International Conference on Distributed Computing Systems (ICDCS)*, 2018.

[143] J. A. Bergstra and C. Middelburg, "ITU-T Recommendation G.107: The E-Model: A Computational Model for Use in Transmission Planning," 2015.

[144] A. Raake, S. Borer, S. M. Satti, J. Gustafsson, R. R. R. Rao, S. Medagli, P. List, S. Göring, D. Lindero, W. Robitza *et al.*, "Multi-model Standard for Bitstream-, Pixel-based and Hybrid Video Quality Assessment of UHD/4K: ITU-T P. 1204," *IEEE Access*, 2020.

[145] I. J. Myung, "Tutorial on Maximum Likelihood Estimation," *Journal of Mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003.

[146] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[147] T. Bauschert, C. Büsing, F. D'Andreagiovanni, A. M. Koster, M. Kutschka, and U. Steglich, "Network Planning under Demand Uncertainty with Robust Optimization," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 178–185, 2014.

[148] K. Dietz, M. Hajizadeh, N. Wehner, S. Geißler, P. Casas, M. Seufert, and T. Hoßfeld, "Certainly Uncertain: Demystifying ML Uncertainty for Active Learning in Network Monitoring Tasks," in *2024 20th International Conference on Network and Service Management (CNSM)*. IEEE, 2024, pp. 1–7.

[149] A. Der Kiureghian and O. Ditlevsen, "Aleatory or Epistemic? Does It Matter?" *Structural Safety*, vol. 31, no. 2, pp. 105–112, 2009.

[150] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[151] E. Hüllermeier and W. Waegeman, "Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods," *Machine Learning*, vol. 110, pp. 457–506, 2021.

[152] M. Seufert, "Fundamental Advantages of Considering Quality of Experience Distributions over Mean Opinion Scores," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2019, pp. 1–6.

[153] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1050–1059.

[154] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[155] A. Thielmann, R.-M. Kruse, T. Kneib, and B. Säfken, "Neural Additive Models for Location Scale and Shape: A Framework for Interpretable Neural Regression Beyond the Mean," *arXiv Preprint arXiv:2301.11862*, 2023.

[156] D. S. Nair, N. Hochgeschwender, and M. A. Olivares-Mendez, "Maximum Likelihood Uncertainty Estimation: Robustness to Outliers," *arXiv Preprint arXiv:2202.03870*, 2022.

[157] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou, "From Distributed Machine Learning to Federated Learning: A Survey," *Knowledge and Information Systems*, vol. 64, no. 4, pp. 885–917, 2022.

[158] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[159] TensorFlow, "TensorFlow Federated," Accessed: July 17, 2024. [Online]. Available: https://www.tensorflow.org/federated

[160] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated Learning in Mobile Edge Networks: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[161] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1759–1799, 2021.

[162] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán, "Decentralized Federated Learning: Fundamentals, State of the Art, Frameworks, Trends, and Challenges," *IEEE Communications Surveys & Tutorials*, 2023.

[163] A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim, "Towards Automatic Concept-Based Explanations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[164] S. Verma, J. Dickerson, and K. Hines, "Counterfactual Explanations for Machine Learning: A Review," *arXiv preprint arXiv:2010.10596*, vol. 2, 2020.

[165] N. Feldhus, A. M. Ravichandran, and S. Möller, "Mediators: Conversational Agents Explaining NLP Model Behavior," *arXiv preprint arXiv:2206.06029*, 2022.

[166] D. Slack, S. Krishna, H. Lakkaraju, and S. Singh, "Explaining Machine Learning Models with Interactive Natural Language Conversations Using TalkToModel," *Nature Machine Intelligence*, vol. 5, no. 8, pp. 873–883, 2023.

[167] H. Shen, C.-Y. Huang, T. Wu, and T.-H. K. Huang, "ConvXAI: Delivering Heterogeneous AI Explanations via Conversations to Support Human-AI Scientific Writing," in *Companion Publication of the 2023 Conference on Computer Supported Cooperative Work and Social Computing*, 2023, pp. 384–387.

[168] N. Feldhus, Q. Wang, T. Anikina, and K. Chopra, "InterroLang: Exploring NLP Models and Datasets through Dialogue-Based Explanations," *arXiv preprint arXiv:2302.02487*, 2023.

[169] N. Wehner, N. Feldhus, M. Seufert, S. Möller, and T. Hoßfeld, "QoEXplainer: Mediating Explainable Quality of Experience Models With Large Language Models," in *2024 16th International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2024.

[170] Z. Hammoudeh and D. Lowd, "Training Data Influence Analysis and Estimation: A Survey," *Machine Learning*, pp. 1–53, 2024.

[171] P. W. Koh and P. Liang, "Understanding Black-Box Predictions Via Influence Functions," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1885–1894.

[172] J. Law, "Robust Statistics—The Approach Based on Influence Functions," 1986.

[173] B. Settles, "Active Learning Literature Survey," 2009.

[174] Z. Liu, H. Ding, H. Zhong, W. Li, J. Dai, and C. He, "Influence Selection for Active Learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9274–9283.

[175] S. Dang, O. Amin, B. Shihada, and M.-S. Alouini, "From a Human-Centric Perspective: What Might 6G Be?" *Authorea Preprints*, 2023.

[176] Y. Liu, Y. Liu, Z. Liu, Y. Liang, C. Meng, J. Zhang, and Y. Zheng, "Federated Forest," *IEEE Transactions on Big Data*, vol. 8, no. 3, pp. 843–854, 2020.

Nikolas Wehner studied computer science at the University of Würzburg, Germany, where he received a Master's degree. In 2018, he worked as a Research Engineer at the Center for Technology Experience at the AIT Austrian Institute of Technology in Vienna, Austria. Since October 2019, he is a doctoral researcher at the Chair of Communication Networks of the University of Würzburg. His interests are user-centric communication networks, focusing on the QoE of Internet applications, and machine learning for networks in general.

Anika Seufert received her Ph.D. in Computer Science from the University of Würzburg, Germany, in 2025. She previously worked as a Research Assistant at the Chair of Communication Networks at the University of Würzburg. Her research interests include user-centric modeling of mobile applications and communication systems, measuring Internet experience from the end-user perspective, and optimizing Quality of Experience.

Tobias Hoßfeld is professor at the Chair of Communication Networks at the University of Würzburg, Germany, since 2018. He finished his PhD in 2009 and his professorial thesis (habilitation) "Modeling and Analysis of Internet Applications and Services" in 2013 at the University of Würzburg, where he was also heading the "Future Internet Applications & Overlays" research group. From 2014 to 2018, he was head of the Chair "Modeling of Adaptive Systems" at the University of Duisburg-Essen, Germany. He has published more than 100 research papers in major conferences and journals, receiving several best conference paper awards, 3 awards for his PhD thesis, and the Fred W. Ellersick Prize 2013 (IEEE Communications Society) for one of his articles on QoE. He is member of the editorial board of IEEE Communications Surveys & Tutorials, Springer Quality and User Experience, ACM SIGMM Records and elected chairperson of the ITG/VDE expert group "Communication Networks and Systems" within the German society of Information Technology (ITG).

Michael Seufert (Senior Member, IEEE) is a Full Professor at the University of Augsburg, Germany, heading the Chair of Networked Systems and Communication Networks. He received the Bachelor's degree (2018) in economathematics and the Diploma (2011), PhD (2017), and Habilitation (2023) degrees in computer science from the University of Würzburg, Germany, and holds the First State Examination degree (2011) in mathematics, computer science, and education for teaching in secondary schools. His research focuses on user-centric communication networks, including QoE of Internet applications, AI/ML for QoE-aware network management, as well as group-based communications.