DIGITWISE: Digital Twin-based Modeling of Adaptive Video Streaming Engagement

Emanuele Artioli
emanuele.artioli@aau.at
Alpen-Adria-Universitaet
Klagenfurt, Kaernten, Austria

Farzad Tashtarian
farzad.tashtarian@aau.at
Alpen-Adria-Universitaet
Klagenfurt, Kaernten, Austria

Christian Timmerer
christian.timmerer@aau.at
Alpen-Adria-Universitaet
Klagenfurt, Kaernten, Austria

ABSTRACT

As the popularity of video streaming entertainment continues to grow, understanding how users engage with the content and react to its changes becomes a critical success factor for every stakeholder. User engagement, i.e., the percentage of video the user watches before quitting, is central to customer loyalty, content personalization, ad relevance, and A/B testing. This paper presents DIGITWISE, a digital twin-based approach for modeling adaptive video streaming engagement. Traditional adaptive bitrate (ABR) algorithms assume that all users react similarly to video streaming artifacts and network issues, neglecting individual user sensitivities. DIGITWISE leverages the concept of a digital twin, a digital replica of a physical entity, to model user engagement based on past viewing sessions. The digital twin receives input about streaming events and utilizes supervised machine learning to predict user engagement for a given session. The system model consists of a data processing pipeline, machine learning models acting as digital twins, and a unified model to predict engagement. DIGITWISE employs the XGBoost model in both digital twins and unified models. The proposed architecture demonstrates the importance of personal user sensitivities, reducing user engagement prediction error by up to 5.8% compared to non-user-aware models. Furthermore, DIGITWISE can optimize content provisioning and delivery by identifying the features that maximize engagement, providing an average engagement increase of up to 8.6 %.

CCS CONCEPTS

• **Human-centered computing** → **User models**; • **Information systems** → **Multimedia streaming**; **Multimedia databases**; *Data analytics*; • **Computing methodologies** → *Ensemble methods.*

KEYWORDS

digital twin, user engagement, QoE, quality of experience, XGBoost, boosted decision tree, HAS, HTTP adaptive streaming

ACM Reference Format:
Emanuele Artioli, Farzad Tashtarian, and Christian Timmerer. 2024. DIGITWISE: Digital Twin-based Modeling of Adaptive Video Streaming Engagement. In *ACM Multimedia Systems Conference 2024 (MMSys '24), April 15–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3625468.3647613

MMSys '24, April 15–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-0412-3/24/04.
https://doi.org/10.1145/3625468.3647613

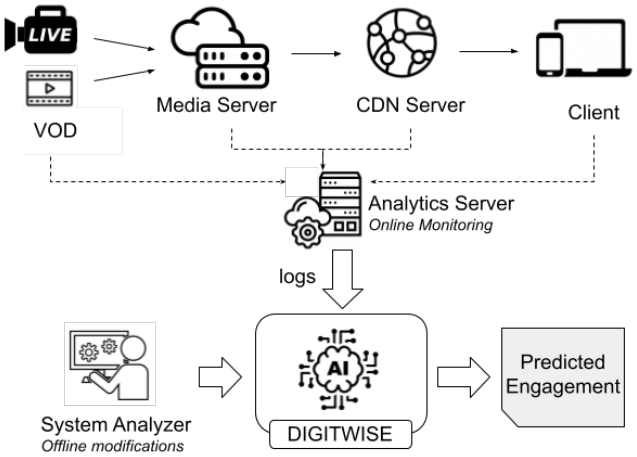

Figure 1: Conceptual architecture of video streaming with DIGITWISE enhancements.

1 INTRODUCTION

Ever since video streaming emerged as one of the main applications of the Internet, with a current share of network traffic of around 65% [1], researchers have been looking for the optimal measure of viewer satisfaction. HTTP Adaptive Streaming (HAS), is the leading approach for video streaming and involves a media server breaking down a video into segments and encoding each segment in various versions distinguished by bitrate and resolution settings [2]. These versions are then organized into a bitrate ladder, sometimes called an encoding ladder, which is essentially a structured list of pairs representing bitrate and resolution. Each client/user, also known as a player, requests the server for a manifest file, containing information about the bitrate ladder and other related data. The delivery of the video occurs segment by segment: when a client requests the next segment, it selects a representation from the bitrate ladder based on an adaptive bitrate (ABR) algorithm. This selection aims to adapt to changing network conditions while balancing conflicting performance goals. For instance, opting for a higher bitrate can enhance video quality but may lead to buffering if the segment does not arrive in time for playback.

User Engagement. The knowledge of how and how much users are impacted by streaming events, such as rebuffering or visual artifacts, i.e., knowledge of user sensitivities, would allow video streaming providers to make better decisions in encoding and delivery.

This in turn would save resources and increase viewer satisfaction and view time, which strongly correlates with revenue [3]. Interestingly, sensitivities are rarely considered in the design of ABR algorithms, implicitly assuming that all users react the same way to the same streaming experience. Research shows this assumption is imprecise and that user sensitivity to artifacts, design choices, and network limitations is fundamental to their unique behavior [4]. However, the discovery and modeling of user sensitivities remain challenging. Initially, user engagement was modeled as uniform and solely dependent on the quality of service (QoS) factors such as throughput, delay, and jitter [5]. Eventually, QoS factors gave way to a more sophisticated measurement of quality of experience (QoE) [6], where users were directly asked about their experience while watching video content, typically via controlled laboratory experiments. Here, user sensitivities were an inextricable part of the evaluation but mixed with every other external factor that cannot be gauged by analyzing session data, i.e., confounding factors, such as interest in the particular content or testing conditions. Instead of separating these confounding factors, initial research focused on modeling the subjective QoE via objective metrics such as video bitrate or number of stalling events [7]. Finally, researchers realized the importance of tracking and modeling actual user engagement expressed in terms of view time [8] or user abandonment likelihood [9]. This measure has the critical advantage of directly impacting video streaming stakeholders such as service and/or content providers. However, collecting user engagement through controlled experiments is a significant challenge since viewers must simulate their watching patterns without choosing the provided content. Therefore, it is harder to single out user sensitivity from confounding factors. Although many studies have collected and utilized large datasets containing key features, none are in the public domain [8, 3, 10]. At the same time, gathering such a dataset is a tremendous effort, which requires users to install plugins on their systems that can record their interactions with streaming services. This is typically collected via crowdsourcing [11] and following many users for weeks or months. Furthermore, it is complicated to track users across multiple devices. An alternative is collaborating with a video streaming provider to collect the necessary data [8].

Digital Twins. Having established the importance of modeling user sensitivities and their impact on engagement, it is necessary to identify the best model for this task. A promising concept for modeling individual user sensitivities is that of digital twins (DTs), i.e., digital replicas of physical twins (PTs) that behave as closely as possible to the real entity for the intents and purposes of the application. This is an already established technology in the manufacturing sector, where it enhances monitoring and forecasting of machines' conditions [12], but it struggles to find a broader scope of application, partly due to the difficulty of producing a general, application-agnostic definition of a DT. [12]. Therefore, the present study leverages DTs that focus on two properties:

- Composition: a DT is not stand-alone but included in a system, together with its PT and an application that leverages the DT to understand or predict properties of the PT.
- Data persistency: a DT keeps a record of its PT states at all times.

In its most refined version, a video streaming engagement DT would replicate the exact behavior of its PT for that session, as is the case for other industries [13]. In other words, it would pause, seek, stop watching, etc., when the user would. This DT is tough to reproduce in a video streaming setting, as fine-grained user behavior is often not related to the session itself but to confounding factors.

In this paper, we introduce DIGITWISE, a machine learning-based system that trains on data collected from every step of the video streaming pipeline, then leverages XGBoost-based DT models to learn unique user sensitivities, and outputs via another XGBoost model, predictions for video streaming engagement. Figure 1 delineates such context. In addition to accurately predicting user engagement, DIGITWISE allows a system analyzer to explore how changing the value of a streaming parameter would impact a group of users' engagement. All that is required is to generate several viewing sessions that differ only in a considered parameter and input these sessions into the DT model, which will output the engagement for each alternative. For example, a video streaming provider interested in the best encoding parameters for their videos can use this system to check the impact of each encoding parameter on their subscribers.

This paper makes the following contributions:

- We present new findings on the impact of viewers' unique sensitivities on their video streaming engagement based on real-world streaming sessions.
- We release our extensive dataset of user streaming sessions both for the reproducibility of the present work and to allow further work in this respect, given the lack of publicly available data[1].
- We develop DIGITWISE, a machine learning-based model that accurately estimates user engagement by leveraging the concept of DTs and user sensitivities.
- We utilize DIGITWISE in conjunction with a cloud-based testbed to compare the impact of different design choices (ABR, bitrate ladder, etc.) on user engagement.

2 RELATED WORK

In this section, we investigate a set of studies that tackled the problem of modeling user experience in video streaming. The current state of the art can be categorized regarding the following features: (*i*) the source of input data: real user sessions vs. controlled laboratory experiment, (*ii*) the employed model: ad-hoc, closed-form functions vs. machine learning models, and (*iii*) the model's objective: QoE vs. engagement. Dobrian et al. [8] provide foundational knowledge regarding the importance of moving toward modeling engagement and how different video streaming features impact it. They collect and analyze a unique dataset of client-side measurements from millions of video viewing sessions, focusing on different quality metrics and engagement levels (see Table 1 for more details). Then, they apply correlation and linear regression of compressed metrics (e.g., number of buffering events, average quality) to model user engagement. Their findings have important implications for content providers to optimize user engagement by reducing buffering, minimizing buffering events for long video-on-demand and live content, and increasing the average bitrate for live content. The authors stress the significance of using measurement-driven insights to improve Internet video delivery. Balachandran et al. [3] expand on the previous paper by using the same dataset to develop a decision tree able to classify user sessions into 10% bins

[1] Available at https://github.com/emanuele-artioli/digitwise

Table 1: Comparing the dataset of DIGITWISE with the state-of-the-art.

	Dobrian et al.[8]	Balachandran et al. [3]	Shafiq et al. [10]	Rodriguez et al. [4]	Robitza et al. [11]	Lebreton et al. [9]	Huang et al. [14]	**DIGITWISE**
startup delay	✓	✓		✓	✓	✓	✓	✓
bitrate	✓	✓	✓	✓	✓	✓	✓	✓
stalls	✓	✓		✓	✓	✓	✓	✓
device type		✓	✓					✓
content type	✓			✓		✓	✓	
video duration	✓		✓		✓			✓
codec			✓	✓		✓		
time of day			✓		✓			✓

Table 2: Comparing the architecture of DIGITWISE with the state-of-the-art.

	Dobrian et al.[8]	Balachandran et al. [3]	Shafiq et al. [10]	Rodrıguez et al. [4]	Robitza et al. [11]	Lebreton et al. [9]	Huang et al. [14]	**DIGITWISE**
input	real sessions	real sessions	real sessions	laboratory test	real sessions	laboratory test	laboratory test	real sessions
model	linear regression	decision tree	decision tree	ad-hoc function	N/A	ad-hoc function	ad-hoc function	XGBoost
output	engagement	engagement	engagement	QoE	engagement	engagement	QoE	engagement

of engagement with 70% accuracy. To achieve this, they analyze potential confounding factors, removing or dividing them into different classes before training, allowing the model to avoid many pitfalls. Shafiq et al. [10] investigate the impact of network factors on user engagement. They build a bagged decision tree-based model capable of accurately predicting whether a user will stop watching before reaching the end of the video by only utilizing the first 10 seconds of network activity. Rodriguez et al. [4] investigate the importance of the user's preference for video content and its impact on a subjective video quality metric. They classify content into soccer, reportage, and tech documentary and assign each user a coefficient per content. These coefficients are added as variables to typical streaming features like rebuffering in a preference factor function that resizes the opinion score of each view. Another valuable contribution of this paper is the usage of video lengths similar to real streams. Robitza et al. [11] use crowdsourcing to collect an extensive dataset about user engagement on multiple popular services, with the vast majority of recorded sessions coming from YouTube. By leveraging the ITU-T P.1203 QoE model [15] and session-level information, they show commonalities and differences among user groups and how these relate to whether users stopped watching at the beginning, middle, or end of a streaming session. Lebreton et al. [9] consider a more extensive set of features and use them to model the user quitting ratio as a function of time. They offer streaming platforms the ability to predict the likelihood of losing a certain number of viewers due to encoding and ABR decisions. They ponder using user sensitivities as a feature to aid in modeling engagement but ultimately limit their scope to model users quitting because of bad quality and not user engagement. Huang et al. [14] realize the potential of leveraging DT-based solutions for video streaming user experience. To develop an optimal ABR algorithm, they extract user QoE factor sensitivities through nonlinear regression. Having obtained the set of factor coefficients for each user, they compute a PQoE model that considers the impact of factors exponentially decreasing with time. Tables 1 and 2 compare the proposed DIGITWISE and related work on two aspects: the completeness of the dataset and the model architecture. The dataset used is the most complete with respect to features that have been shown to impact user engagement. Model-wise, DIGITWISE does not include the content type and codec simply because those features were missing from the training data due to our data provider's privacy policies. A more complete dataset would not require changes in the DIGITWISE architecture and would allow a more robust modeling of user engagement.

3 DIGITWISE

Figure 2 illustrates the DIGITWISE architecture, including the two main planes: (*i*) *sensitivity plane* and (*ii*) *engagement plane*.

The **sensitivity plane** consists of four components: input database, data processing, per-user DT, and sensitivity database. The process begins by collecting data from Bitmovin Analytics [16] into a database. In this study, the dataset consists of one row per session event, meaning a new row of data is collected each time a user starts, plays, pauses, seeks, or quits the session or an internal event happens during the streaming that impacts QoE (e.g., bitrate change, stall). This new row of data includes information about time, location, user identifier (ID), video ID, session ID, device type, bitrate, etc. Such a rich dataset allows us to draw accurate conclusions about the impact of many factors on user engagement, isolate the effects of confounding factors, and verify the results of changing the values of a factor. Subsequently, the data undergo a processing task aimed at: (*i*) eliminating sessions, i.e., the rows relating to the

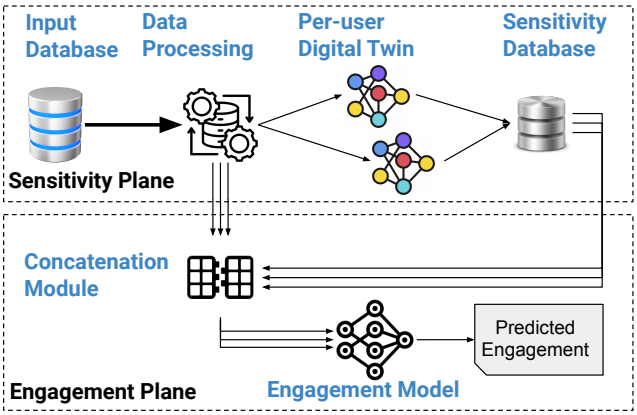

Figure 2: DIGITWISE architecture.

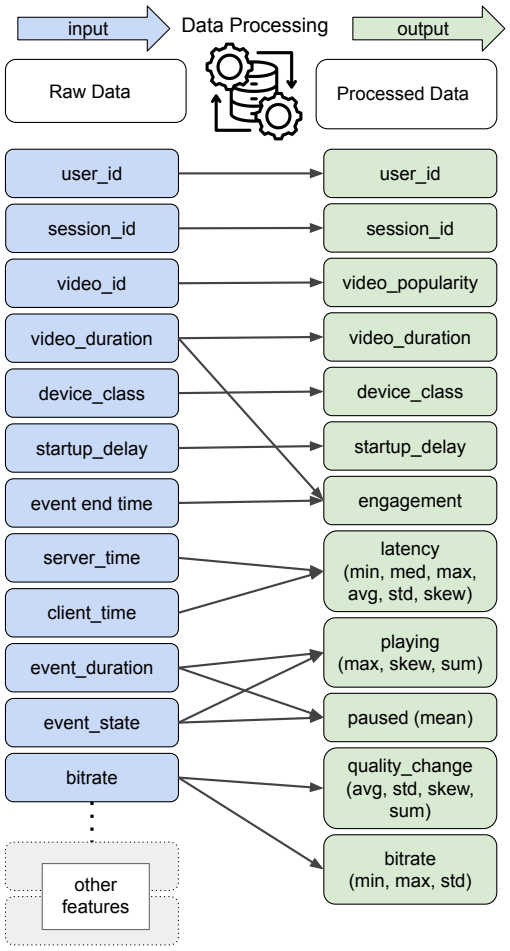

Figure 3: Inputs and outputs of the Data Processing component.

same viewing experience, with collection errors or missing data, (*ii*) engineering new useful features such as video popularity or engagement, (*iii*) consolidating all data related to the same session into a single row through the calculation of aggregate metrics, (*iv*) selecting users with enough data to enable reliable models, and (*v*) performing feature selection. Figure 3 shows the input and output parameters to and from the data processing component. The processed data is then utilized to train a set of initial models known as DTs. Each DT is trained on a subset of sessions by a specific user, enabling the models to learn individual user sensitivities. Once the models are effectively trained, information gain [17] and permutation importance [18], which will be explained in the following subsection, are employed to extract user sensitivities. These sensitivities, which provide valuable insights into user engagement, replace the need for user IDs since they contain information regarding what differentiates one user from others, i.e., its sensitivities becoming available in the *sensitivity database*.

The **engagement plane**, consisting of two main components (i.e., concatenation module and engagement model), uses the provided data by the sensitivity plane to estimate user engagement accurately. The *concatenation module* combines the provided sensitivity features with the selected features associated with the same user from the data processing component. The output of the concatenation module is fed into a unified *engagement model*. The engagement model is trained on all user-engineered features and their sensitivities. Consequently, it can leverage similarities among all users while providing distinct engagement predictions based on the individual user. In the following, we dive into the details of the sensitivity and engagement planes.

3.1 Sensitivity Plane

Input Database. Data about user sessions, collected from player logs by the analytics server, are stored in a database. Each record includes more than 100 raw features that capture all events occurring in the player, whether caused by the user or internal player processes (e.g., ABR algorithm). These features encompass information such as the time of day the video was streamed, what content delivery network (CDN) server was used to deliver it, what user and device streamed it, and when events like bitrate switch and stalls occurred.

Data Processing. Figure 4 shows the DIGITWISE pipeline consisting of various tables given to/generated by computational components. The *Raw Data* table shows the prototype of raw data including $3 + k$ features: `user_id`, `video_id`, `session_id`, and k extra raw features. The first step in our preprocessing pipeline regards data cleaning. Many collection errors and incongruent information plague the raw features, therefore we set up a series of commands to identify and fix these issues:

- Standardize null values into a value recognized by the models;
- Remove users with watching patterns incompatible with real, single people, i.e., watching the same video hundreds of times;
- Remove users with less than 50 viewing sessions, to avoid training DTs on too little data;
- Remove users who watched less than five different videos, to avoid picking up on patterns that are specific to the video and not the user;

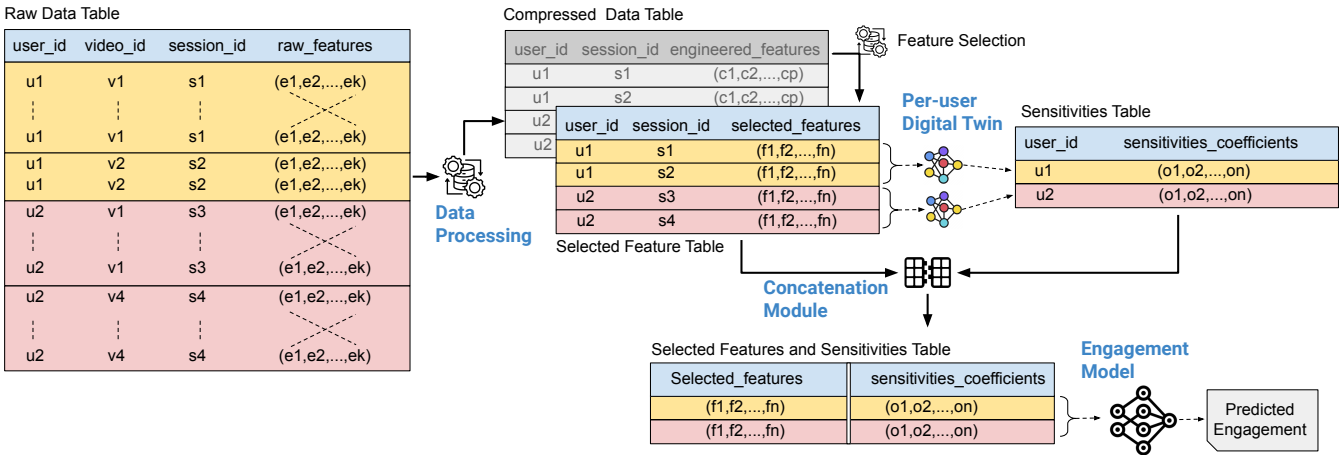

Figure 4: DIGITWISE pipeline.

- Video duration is not consistent along each video id, so it is standardized to each video's max video duration;
- Remove sessions that encountered errors, since we only need sessions that ended because of the user's choice;
- Remove sessions with negative event end time, which would make it impossible to gauge engagement;
- Remove sessions that were watched for less than 10 seconds;
- Remove sessions longer than 24 hours since they are likely to be 24/7 live channels whose engagement we cannot compare to regular videos;
- Event duration is sometimes negative, thus, remove affected sessions;
- Sometimes videotime end is greater than video duration, which is impossible, thus, remove affected sessions.

We also take care of engineering several features to allow the models to extract as much information about the viewing sessions as possible:

- Convert time into hour of day, as time of day impacts user engagement [19];
- Convert video id into a video popularity score, that reflects how many times each video was watched, which is another feature impacting user engagement [19];
- The type of event (play, pause, seek, etc.) gets condensed into a more precise and less redundant space;
- Device class (TV, laptop, phone, etc.) gets remapped into screen size, to leverage its order;
- Add bitrate switches as the difference between the bitrate in the previous event and the current event;
- Add latency between client and analytics server as the difference between the timestamp added at client side before sending data and the one collected at server side as data arrives;
- Engagement is derived as maximum videotime end divided by video duration, in other words the furthest point in the video the user reached before quitting.

Data Compressing. The analytics server collects data from one row per event, meaning that each time the player receives a playback request, starts playing, changes bitrate, pauses, etc., one new row is appended to the data about that user, video, and session. This data shape is valuable for time-sensitive applications such as an online DT predicting real-time engagement. Our current goal of offering video streaming service providers insight into their design choices will likely not benefit from the added complexity of developing a time-sensitive model to handle event-based data. Therefore, the raw data needed to be compressed from one row per event into one row per user/session via aggregate metrics such as averages and standard deviations. As shown in Figure 4, the occurred events during two sessions of user $u1$ are compressed into two rows in *Compressed Data* table, each of which has $2 + p$ features including `user_id`, `session_id`, and p `engineered_features`, where $k < p$. Compressing multiple rows of raw data into one row inevitably reduces some parts of information due to the order and positions of events that significantly impact user engagement. For example, this is clear when considering the scenario of two users playing the same video simultaneously. The first user experienced the first minute of playback with frequent stalls and is now at minute 10 after 9 minutes of smooth playback. The second user experienced the first 9 minutes without problems and is now overwhelmed by frequent stalls for a minute. Which user do we assume is the most likely to stop engaging with the content in the immediate future? This problem is highlighted in [9], and we offer a simple but efficient solution as follows. To retain information on whether certain events were concentrated more towards the playback's beginning or end, we compute the skewness metric for features that benefit from it, such as bitrate switches and stalls. In the example above, while the stalls' mean and standard deviation are the same, the skewness is positive for the first user with initial stalls and negative for the second where the stalls occur at the end.

User Selection. To avoid models that are skewed towards a particularly prevalent engagement value, we separate sessions into ten engagement bins (meaning, we group sessions with engagement from 0 to 10%, from 10% to 20%, etc.) and sample these groups so that they have the same number of sessions. Then, we count how many sessions are left for each user and remove each user with less than 100 sessions to avoid DTs from training on too few data points to be relied upon. Once filtered, each user's sessions are split, 80% make up a training set, the remaining a test set.

Feature Selection. Another important step of the processing is feature selection. Many of the p engineered features show a high correlation. Although not directly harmful to model predictions, a high number of features slows down training and hinders the production of feature importance scores. Therefore, we devise a semi-automatic feature selection comprising the following steps:

- We train a Random Forest Regressor [20] model on the compressed dataset.
- From the trained model, we extract feature importance computed as the mean decrease in entropy [17].
- We calculate the Spearman rank correlation [21] coefficient matrix for the feature set.
- We sum the correlation coefficients for each feature to compute a rough indication of how much each feature is correlated to all others.
- We divide the previously obtained feature importance by the sum of correlation coefficients just computed to obtain a new measure of feature importance that is penalized when other features are strongly correlated to it.
- For better interpretation, we normalize the coefficients.

The features with the highest coefficient are important and have low correlations among themselves. We can select the feature set that maximizes the performance to computing cost tradeoff by optimizing a single threshold for feature importance. Furthermore, we removed any feature that might compromise the customer's or its users' privacy. The `selected feature table` in Figure 4 shows $2 + n$ columns including `user_id`, `session_id`, and n selected features, where $n < p$. It is important to note that determining an appropriate threshold value is a delicate and fundamental tradeoff: a high threshold value will only allow the few most significant features, resulting in a fast, simple, and interpretable model. However, the model runs the risk of missing out on the performance gain that less important features might have brought. A low threshold allows the model to extract as much information as possible from the data but increases its computational requirements and might even hinder its performance in the case of collinear features, as explained in the previous section.

Per-user DT. For each user's processed session data, e.g., two rows associated with user $u1$ in the *selected feature table* in Figure 4, we train a model, taking the role of a DT and modeling the user features into an engagement prediction. In other words, we need to determine the sensitivity coefficient value for each user's selected feature based on the calculated engagement value from the raw data (see the `sensitivities table` in Figure 4). For this task, we use XGBoost [22], an ensemble algorithm that leverages boosted decision trees to achieve high performance by focusing its training on the iteratively hardest predictions. Aside from its ease of interpretation and fast convergence, literature has shown how XGBoost is often among the best predictors of tabular data [23]. It is worth mentioning that the optimal hyper-parameters for each user's model, such as the number of trees to train and the number of features each tree had at its disposal, are identified through halving grid search [24]. This technique generates many candidate models with randomly picked sets of hyper-parameters, trains them on limited amounts of data, evaluates their performance via cross-validation, discards the ones with poor performance, and trains the survivors on more data, iterating until only the best model is left. Having obtained a reliable level of performance, these models are scrutinized to extract each feature's impact on engagement. XGBoost natively offers the importance of features via information gain [17].

3.2 Engagement Plane

Concatenation Task. Having obtained each feature's importance for each user, we use this information about user sensitivities as an embedding of that user, substituting the sensitivities to the user ID (see `selected feature and sensitivities table` in Figure 4). Category embedding [25] is a typical technique in machine learning that aims at compacting a high-cardinality categorical feature, i.e., user ID, into a low-dimensional, numerical vector where each scalar represents that user's position concerning other users in a particular dimension. This step allows machine learning models to extract meaning from categorical features, which would be hard without converting them into embedded numerical dimensions. These dimensions are usually hard to interpret and are discovered by the embedding algorithm upon training. On the contrary, each dimension of our user sensitivity vector has a precise meaning, i.e., it represents the sensitivity of that user to a specific selected feature.

Engagement Model. These sensitivities, which provide valuable insights into user engagement, replace the need for user IDs since they contain all the information regarding what differentiates one user from others. Once the user sensitivities have been concatenated with their respective sessions, these enhanced sessions are fed as input into a unified XGBoost model. This model is optimized via halving grid search [24], similar to the sensitivity models. This way, we get the best of both worlds: a sufficient amount of sessions to extract the general behavior (similar to a degree in each user) and the digital-twin improvement upon it by using specific information about each user.

4 PERFORMANCE EVALUATION

To evaluate the performance of DIGITWISE in various scenarios and also compare it with state-of-the-art approaches, we set up a Python Jupyter notebook to run on an Ubuntu server sporting an Intel Xeon Gold 5218 CPU @ 2.30GHz (128 cores), an NVIDIA Quadro GV100 GPU (32 GB VRAM, 5,120 CUDA Cores), 384 GB of RAM, and 9.6 TiB of RAID5 SSD storage. The code is available on GitHub, together with the data used for the training[2].

4.1 Dataset

In this study, we focus on one dataset consisting of a real world video streaming services related to car race events between May and August 2023. This translated into a dataset with 102,233,138 rows and 102 features, describing events from 3,261,381 unique viewing sessions of 50,942 real users watching 3,181 different videos. Such a varied dataset contains much information deemed irrelevant to the current study. Therefore, only the features shown in Figure 3 are effectively used by DIGITWISE. Vice versa, many features that are useful for engagement modeling are absent from the available data. Notable examples include the codec, any information about

[2]Available at https://github.com/emanuele-artioli/digitwise

the session's environment such as viewing distance or light conditions, user demographics, etc. Furthermore, the data relates to a single content provider with a narrow scope (car races). Therefore, considerations about content type and impact on the user base are impossible. Including greater variability and additional features will likely improve DIGITWISE's performance, and future efforts will be made in this direction.

4.2 Experimental Scenarios

We define the following scenarios to assess DIGITWISE's performance under varying design parameters. Additionally, we will compare it with three alternative approaches. Finally, we will demonstrate how DIGITWISE can assist video streaming service providers to fine-tune their streaming service parameters.

Scenario 1: *Comparing the impact of XGBoost and Multi-Layer Perceptron (MLP) on the DIGITWISE performance.* We identified two potential model architectures for DIGITWISE, namely XGBoost and MLP [26]. These represent the two leading approaches with regard to machine learning on tabular data, respectively, Decision Tree-based models and Neural Networks. Both models are optimized via halving grid search, and their training time and mean average error results will be compared. We are interested in a model that is not only precise in its prediction of engagement, but can also offer accurate predictions early in the session. Customers would benefit from knowing the user engagement after only a few minutes into the session, and be thus able to implement changes in the session before the user stops engaging. Therefore, each model is trained multiple times, each time with a different training dataset where only session events happening in the first seconds or minutes are included.

Scenario 2: *The impact of extracting sensitivities coefficients on the DIGITWISE performance.* After identifying the optimal data modeling architectures, we assess the influence of user sensitivities. This evaluation involves training the selected model architectures on the same data twice: first without considering user sensitivities and then with the inclusion of user sensitivities. The discrepancy in performance between these two models reflects the additional information gained from incorporating the sensitivity coefficients.

Scenario 3: *Determining the appropriate threshold value for the feature selection module.* We mentioned before how a single threshold value is responsible for the feature selection step. By tweaking this value, we can influence which features will make their way into the training data for DIGITWISE.

Scenario 4: *Assessing the performance of DIGITWISE compared to the state-of-the-art approaches.* In this scenario, we will compare the performance of DIGITWISE with the following state-of-the-art models over various time horizons and threshold values:

- Balachandran et al. [3] sets up its Decision Tree-based model for a classification task of predicting user engagement in a 10-bin fashion using the whole session data. They obtain an accuracy of 70%. In other words, 7 out of 10 times, the model classifies the real session engagement within a margin of error of 10%.
- Shafiq et al. [10] developed a binary model capable, with 87% accuracy, of predicting whether the user will download the whole video after only seeing the first 10 seconds of network data.

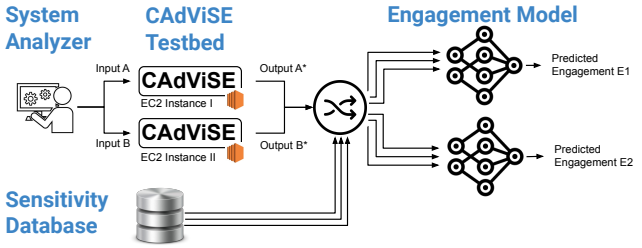

Figure 5: Service tuning architecture.

- Finally, we compared DIGITWISE with [9], whose model groups the data by video and outputs the percentage of users that will stop watching each video "midway through". They then compute the Pearson Correlation Coefficient [21].

Scenario 5: *Changing streaming parameters and measuring its impacts on the user engagement by DIGITWISE.* In this scenario, we will show a useful application of the DIGITWISE model in fine-tuning the streaming parameters by measuring user engagement. To this end, we leverage CAdViSE [27], a cloud-based adaptive video streaming evaluation framework for the automated testing of adaptive media players. CAdViSE simulates content streaming using Amazon Web Services (AWS). Its main goal is to offer a testbed for comparing different streaming parameters (e.g., ABR algorithms, bitrate ladder, etc.) under the same network and player conditions, but it is used in this context to produce realistic session data whose parameters can be carefully determined.

Figure 5 shows the architecture used for Scenario 5. For example, let's assume a system analyzer is interested in the impact that a particular ABR algorithm has on a set of users. They can then generate CAdViSE sessions that share the same network trace, content streamed, segment size, etc., and differ only in their ABR of choice. By adding the interested users' sensitivities to these simulated sessions and using the combination as input to the trained engagement model, the system analyzer can obtain multiple engagement predictions for each user, one prediction per ABR, and their difference is the expression of the impact of said ABR on that user's engagement.

To benchmark this scenario, we generate 100 CAdViSE sessions for each combination shown in Table 3. We concatenate the generated sessions with user sensitivities drawn randomly from the available users and run the engagement model to predict an engagement value for each session. Then, we compute aggregate metrics such as the average engagement for each value of the initially chosen feature.

This tells us the potential for that design choice: if we were hypothetically using the value for each feature that is shown to have the lowest engagement for the chosen users, how much could the engagement improve by switching to the value shown to have the highest engagement?

4.3 Results and Analysis

Scenario 1.

Figure 6 and Figure 7 show the cumulative times for the training of: (*i*) the sensitivity models, (*ii*) the benchmark models, i.e., the models that train on the whole data set without the sensitivity

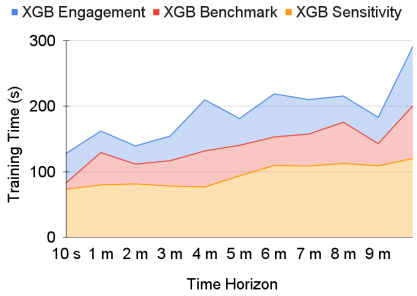

Figure 6: XGBoost training time.

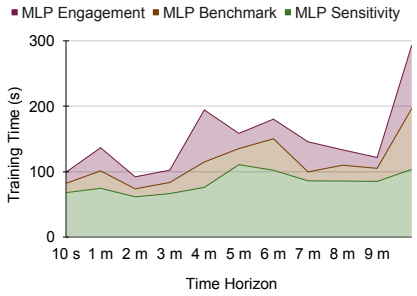

Figure 7: MLP training time.

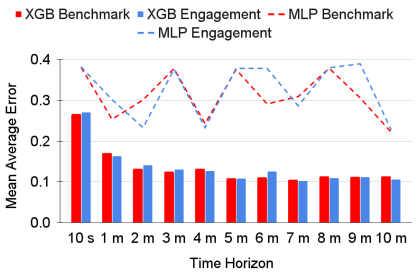

Figure 8: MAE comparison.

Table 3: Measuring the engagement over various CAdViSE sessions distribution.

Segment size (s)	ABR	Network trace	Mean	Std.	Min	Median	Max
1	L2A	AmazonFCC	0.638	0.123	0.471	0.720	0.788
1	L2A	Cascade-20	0.616	0.127	0.455	0.576	0.785
1	L2A	Cascade-5	0.611	0.142	0.415	0.575	0.786
1	L2A	Constant (16 Mbps)	0.631	0.124	0.403	0.685	0.763
1	L2A	LTE [28]	0.634	0.132	0.425	0.576	0.785
2	Bola	AmazonFCC	0.597	0.145	0.411	0.567	0.785
2	Bola	Cascade-20	0.674	0.120	0.453	0.744	0.785
2	L2A	AmazonFCC	0.601	0.140	0.423	0.569	0.783
2	L2A	Cascade-20	0.588	0.138	0.412	0.567	0.785
2	L2A	Cascade-5	0.627	0.136	0.428	0.576	0.786
2	L2A	Constant (16 Mbps)	0.638	0.137	0.451	0.744	0.771
2	L2A	Constant (4 Mbps)	0.610	0.123	0.468	0.554	0.757
2	L2A	LTE	0.554	0.127	0.398	0.565	0.687
2	LoL+ [29]	Cascade-5	0.621	0.145	0.427	0.572	0.786
2	LoL+	LTE	0.620	0.137	0.423	0.569	0.787
2	Throughput [30]	AmazonFCC	0.610	0.139	0.400	0.573	0.784
2	Throughput	Cascade-20	0.634	0.127	0.453	0.722	0.770

features, and (*iii*) the engagement models for XGBoost and Multi-Layer Perceptron. The architectures require similar times to train, which is important to allow for a fair performance comparison. Furthermore, the figures also show how the time required to train depends not on the time horizon but on the model under training: for both architectures, most time is spent training the sensitivity model. Considering that one sensitivity model is required for each user while the same engagement model is used for all users, one can easily understand such behavior.

Regarding the Mean Average Error (MAE), as shown in Figure 8, XGBoost obtains a much lower MAE when compared to MLP, regardless of available data. Furthermore, it shows a more consistent behavior, steadily improving its performance as more data is available, then settling around the 7-minute time-horizon mark, likely for a combination of two factors: diminishing returns of a longer time horizon and difficulty in lowering an already low MAE. This is not true for MLP, especially in its engagement phase. Further evaluation might shed light on this peculiar behavior of MLP, but is deemed irrelevant for the current study, which accepts XGBoost as the superior model for further performance evaluation and practical applications.

Scenario 2.

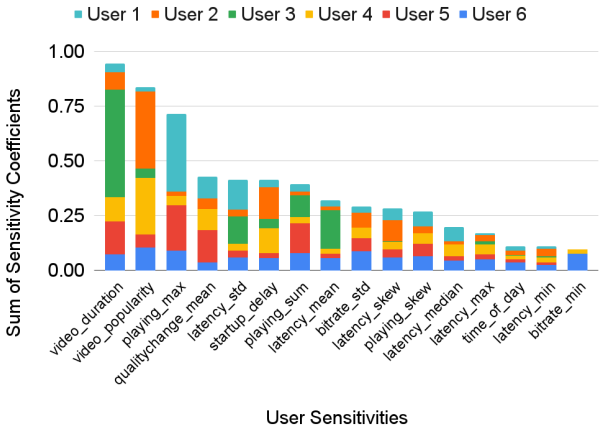

Figure 9: Model sensitivities.

Figure 8 reports another important metric. We benchmark our model, i.e., XGB Engagement, against a model with the same architecture, trained on the same streaming session, and only lacking the information about user sensitivities, i.e., XGB Benchmark. We show how personal user sensitivities can reduce the mean average error by up to 5.8% (for the model with a 10-minute time horizon) and, therefore, prove to be a determining factor in users' willingness to continue watching. Moreover, Figure 9 shows the sensitivity of users to each training feature. For clarity, only users with a high number of sessions were selected. User sensitivities show a clear trend, where most users typically share the most important features. Still, a significant degree of individuality can be observed. Video duration and popularity are the most important factors in determining the final user engagement. This is not surprising, especially in the case of video duration, as a very long video requires much more interest from a user to be completed than a short one Furthermore, long videos have a much higher chance of being affected by network, external situations, and several different confounding factors that can cause a premature interruption in the session. It is important to note how this result is directly caused by how the

engagement metric was measured: if engagement were measured as time elapsed before quitting, instead of a percentage of video duration, the first few minutes of a video would have the same chance for interruption regardless of duration. However, this would have rendered our study not comparable with the state-of-the-art and would have introduced other issues. The most critical issue would have been the inversion of the aforementioned trend since a short video would be penalized by not having the chance to reach the watch time of a long one. Therefore, while not perfect, the chosen engagement metric was nonetheless deemed correct. Another important observation to draw from Figure 9 is how, while being by far the most important feature overall, no single feature is the most important for most users. This is crucial as it represents the main reason for the present study. It demonstrates how users noticeably differ in their importance of video streaming features and how different factors influence their behavior.

Scenario 3.

To identify the optimal feature selection threshold, we trained the model at various threshold levels, from very low, where almost every available feature was included, to very high, where only features of the utmost importance were selected. The results are shown in Figures 10- 12 . As the figures show, very high thresholds do not allow enough features to offer correct predictions, while very low thresholds do not incur the aforementioned risk of collinearity. This is because the XGBoost model's architecture, like every decision tree-based architecture, is resistant to feature collinearity and can disregard unimportant features and focus on the important variability. However, Figure 10 demonstrates how the training time increases with lower thresholds because of the higher number of features, and the model interpretation is likewise harder.

Scenario 4.

Figures 14-13 show how DIGITWISE compares against three benchmarks:

- In the benchmark by Balachandran et al. [3], DIGITWISE is able to approach the target accuracy on occasion but tends towards a slightly lower level.
- Our data does not include information about downloaded segments. Vice versa, [10] only had access to network data and could not distinguish between a video downloaded but not watched in full. Therefore, a perfect comparison is not possible. To benchmark against them, we needed to identify a threshold of engagement above which to consider a session to be the equivalent of a fully downloaded. Two extreme threshold values were considered: 50%, i.e., each video watched for at least half is in the downloaded class. This offered balanced classes, as we sampled our dataset uniformly over engagement. However, in practice, a playback reaches 50% way sooner than being completely downloaded, so this threshold is poorly adapted to the reality of streaming. Vice versa, a threshold of 100%, where only the videos with an engagement of almost 1 were considered for the downloaded class, is adherent to the reality of video streaming, but highly imbalanced. Therefore, we chose a 70% threshold as a middle ground among these extremes. Under this hypothesis, DIGITWISE achieves a 53% accuracy at the 10-second time horizon but improves steadily as more data is available, matching the benchmark at the 2-minute mark and reaching an accuracy of 93%.
- Lebreton et al. [9] does not provide a quantitative value for quitting "midway through". Therefore, we assume that the value is 50% and consider sessions with an engagement higher than that as being in one class and sessions with a lower engagement to comprise the other class. DIGITWISE reaches 0.8 PCC with 3 minutes of data and improves slowly from there, remaining slightly lower than the benchmark of 0.87 PCC.

It should be noted that DIGITWISE was not re-trained with these new target metrics. Instead, we used the same regression model to obtain engagement predictions, then engineered these predictions into the same shape as the ones from the benchmark to calculate DIGITWISE performance. Likely, a model trained on the final prediction shape and using the same scoring metric (instead of the root mean squared error scoring used to train DIGITWISE) would achieve higher results in a specific benchmark, but the way the benchmark has been conducted was considered a more fair assessment of DIGITWISE capabilities.

Scenario 5. AWS EC2 machines are utilized to generate CAdViSE sessions used in the application of the model. For the present scope, we consider three features, i.e., segment size, network trace, and ABR, that directly impact one or more of the engagement model's features so that even if the model does not train directly on them, it is still possible to gauge their contribution. This is a limitation of the current study, which ideally would include each feature of interest, but these were not present in the available data, and their inclusion is left for future work. The features and distribution of sessions for each combination of features are presented in Table 3.

- ABR: The highest difference is seen with a segment size of 2 seconds and network trace Cascade-20 [31](the value of x in Cascade-x indicates the duration of fixed bandwidth). For this combination, moving from L2A [32] to Bola [33] increases the average engagement by 8.6%.
- Network trace: The most highly represented combination is a segment size of 2 seconds, and Bola ABR. For this combination, moving from AmazonFCC [34] to Cascade-20 [31] yields an average engagement increase of 7.7%.
- Segment size: The most highly represented combination is an ABR L2A and a network trace AmazonFCC. For this combination, moving from a segment size of 2 seconds to 1 second increases the average engagement by 3.8%.

Therefore, in the worst-case scenario of a video streaming platform set up with the worst parameters, modifying these parameters to those suggested by DIGITWISE would lead to an increase in engagement of more than 8.6%. Worthy of note is the sometimes small engagement increase when comparing two very different network traces. For example, moving from low bandwidth to high bandwidth increases engagement by 2.7%, which might seem like a small improvement for such a high bandwidth difference. However, this is consistent with Figure 9, where the features impacted by the network trace do not appear in the top most important features. Thus, the choice of user is again proven to be significant: while better network conditions may prove critical for certain users, they are not for everyone.

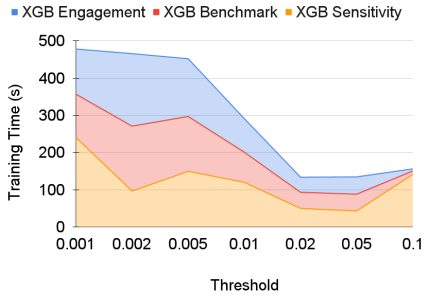

Figure 10: XGBoost training time over different threshold values.

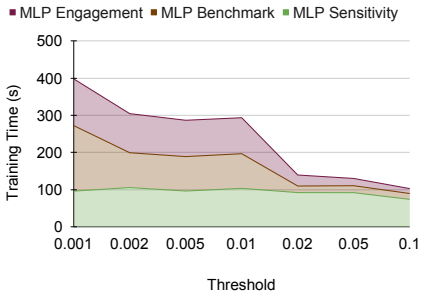

Figure 11: MLP training time over different threshold values.

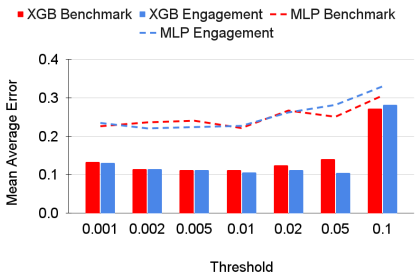

Figure 12: MAE comparison over different threshold values.

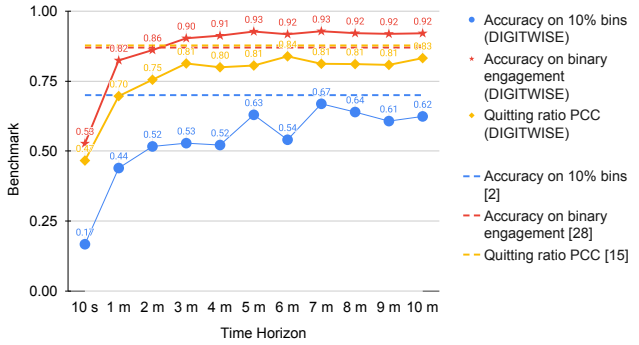

Figure 13: Comparing the performance of DIGITWISE with benchmarks [3, 10, 9] over different time horizon values.

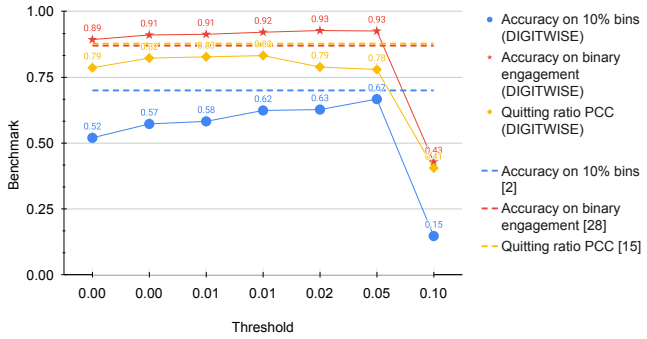

Figure 14: Comparing the performance of DIGITWISE with benchmarks [3, 10, 9] over different threshold values.

5 CONCLUSION

In conclusion, this paper demonstrated how user sensitivities, while often overlooked, are an important factor in engagement considerations and how, by leveraging them, DIGITWISE is able to significantly improve engagement modeling and predictions. Specifically, the Mean Average Error in engagement predictions decreased by up to 5.8% when including user sensitivities in the model. Furthermore, DIGITWISE is flexible and can match different state-of-the-art algorithms in their respective benchmarks without training on any specific task. This flexibility allows DIGITWISE to serve various applications, for example, by adding real users behaviors to simulated sessions and gauging the engagement improvements of specific design choices. While not exhaustive, the feature set available for this experiment allowed us to suggest several design improvements that could increase user engagement by up to 8.6%. The DIGITWISE approach offers a valuable framework for enhancing adaptive video streaming by personalizing the user experience. With a more comprehensive dataset and/or models capable of time-sensitive predictions, DIGITWISE's performance can be pushed even further.

REFERENCES

[1] Sandvine Holdings UK Limited. 2022. Global internet phenomena report. Retrieved June 15, 2023 from https://www.sandvine.com/global-internet-phenomena-report-2022.

[2] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. 2018. A survey on bitrate adaptation schemes for streaming media over http. *IEEE Communications Surveys & Tutorials*, 21, 1, 562–585.

[3] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. 2013. Developing a predictive model of quality of experience for internet video. *ACM SIGCOMM Computer Communication Review*, 43, 4, 339–350.

[4] Demóstenes Z Rodrıguez, Renata L Rosa, Eduardo A Costa, Julia Abrahão, and Graca Bressan. 2014. Video quality assessment in video streaming services considering user preference for video content. *IEEE Transactions on Consumer Electronics*, 60, 3, 436–444.

[5] Victor Firoiu, J-Y Le Boudec, Don Towsley, and Zhi-Li Zhang. 2002. Theories and models for internet quality of service. *Proceedings of the IEEE*, 90, 9, 1565–1591.

[6] Yanjiao Chen, Kaishun Wu, and Qian Zhang. 2014. From qos to qoe: a tutorial on video quality assessment. *IEEE Communications Surveys & Tutorials*, 17, 2, 1126–1165.

[7] Werner Robitza et al. 2018. Http adaptive streaming qoe estimation with itu-t rec. p. 1203: open databases and software. In *Proceedings of the 9th ACM Multimedia Systems Conference*, 466–471.

[8] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. 2011. Understanding the impact of video quality on user engagement. *ACM SIGCOMM computer communication review*, 41, 4, 362–373.

[9] Pierre Lebreton and Kazuhisa Yamagishi. 2020. Predicting user quitting ratio in adaptive bitrate video streaming. *IEEE Transactions on Multimedia*, 23, 4526–4540.

[10] Muhammad Zubair Shafiq, Jeffrey Erman, Lusheng Ji, Alex X Liu, Jeffrey Pang, and Jia Wang. 2014. Understanding the impact of network dynamics on mobile video user engagement. *ACM SIGMETRICS Performance Evaluation Review*, 42, 1, 367–379.

[11] Werner Robitza, Alexander M Dethof, Steve Göring, Alexander Raake, André Beyer, and Tim Polzehl. 2020. Are you still watching? streaming video quality and engagement assessment in the crowd. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 1–6.

[12] Fei Tao, He Zhang, Ang Liu, and Andrew YC Nee. 2018. Digital twin in industry: state-of-the-art. *IEEE Transactions on industrial informatics*, 15, 4, 2405–2415.

[13] Aitor Moreno, Gorka Velez, Aitor Ardanza, Iñigo Barandiaran, Álvaro Ruíź de Infante, and Raúl Chopitea. 2017. Virtualisation process of a sheet metal punching machine within the industry 4.0 vision. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 11, 2, 365–373.

[14] Xinyu Huang, Conghao Zhou, Wen Wu, Mushu Li, Huaqing Wu, and Xuemin Shen. 2022. Personalized qoe enhancement for adaptive video streaming: a digital twin-assisted scheme. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 4001–4006.

[15] Alexander Raake, Marie-Neige Garcia, Werner Robitza, Peter List, Steve Göring, and Bernhard Feiten. 2017. A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1. In *Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, Erfurt, (May 2017). ISBN: 978-1-5386-4024-1. DOI: 10.1109/QoMEX.2017.7965631.

[16] [n. d.] Track, monitor and analyze your streams in real-time with Online Video Analytics. [Online] Available: https://bitmovin.com/video-analytics. ().

[17] Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27, 3, 379–423.

[18] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. 2010. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26, 10, 1340–1347.

[19] Yishuai Chen, Yong Liu, Baoxian Zhang, and Wei Zhu. 2014. On distribution of user movie watching time in a large-scale video streaming system. In *2014 IEEE International Conference on Communications (ICC)*, 1825–1830. DOI: 10.1109/ICC.2014.6883588.

[20] Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE, 278–282.

[21] Jerome L Myers, Arnold D Well, and Robert F Lorch Jr. 2013. *Research design and statistical analysis*. Routledge.

[22] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: a scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

[23] Manuel Fernández-Delgado, Manisha Sanjay Sirsat, Eva Cernadas, Sadi Alawadi, Senén Barro, and Manuel Febrero-Bande. 2019. An extensive experimental survey of regression methods. *Neural Networks*, 111, 11–34.

[24] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13, 2.

[25] Lütfi Kerem Şenel, Ihsan Utlu, Veysel Yücesoy, Aykut Koc, and Tolga Cukur. 2018. Semantic structure and interpretability of word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26, 10, 1769–1779.

[26] Berndt Müller, Joachim Reinhardt, and Michael T Strickland. 1995. *Neural networks: an introduction*. Springer Science & Business Media.

[27] Babak Taraghi, Anatoliy Zabrovskiy, Christian Timmerer, and Hermann Hellwagner. 2020. Cadvise: cloud-based adaptive video streaming evaluation framework for the automated testing of media players. In *Proceedings of the 11th ACM Multimedia Systems Conference*, 349–352.

[28] J. Van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters*, 20, 11, 2177–2180.

[29] May Lim, Mehmet N Akcay, Abdelhak Bentaleb, Ali C Begen, and Roger Zimmermann. 2020. When They Go High, We Go Low: Low-Latency Live Streaming in dash.js with LoL. In *Proceedings of the 11th ACM Multimedia Systems Conference*, 321–326.

[30] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. 2011. Rate adaptation for adaptive http streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*, 169–174.

[31] Abdelhak Bentaleb, May Lim, Mehmet N Akcay, Ali C Begen, and Roger Zimmermann. 2021. Common Media Client Data (CMCD) Initial Findings. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 25–33.

[32] Theo Karagkioules, Rufael Mekuria, Dirk Griffioen, and Arjen Wagenaar. 2020. Online Learning for Low-Latency Adaptive Streaming. In *ACM MMSys*, 315–320.

[33] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2020. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. *IEEE/ACM Transactions on Networking*, 28, 4, 1698–1711.

[34] [n. d.] Raw Data — Measuring Broadband America. [Online] Available: https://www.fcc.gov/reportsresearch/reports. ().

Received 20 September 2023; revised 12 March 2024; accepted 5 June 2024