# Conversion rate optimization in e-commerce: using machine learning to identify website satisfaction in clickstream patterns

Dragos Tomescu
STUDENT NUMBER: U1278151

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee:

DR GIOVANNI CASSANI
DR DREW HENDRICKSON

Tilburg University
School of Humanities and Digital Sciences Department of Cognitive Science & Artificial Intelligence Tilburg, The Netherlands
June 2020

## Table of Contents

# Conversion rate optimization in e-commerce: using machine learning to identify website satisfaction in clickstream patterns

Dragos Tomescu

*Despite website satisfaction becoming paramount in an increasingly digitalised world and a mounting need for better tools which allow practitioners to understand complex clickstream behaviour, current applications fail to capture the link between online clickstream behaviour and website satisfaction. To address this gap, this study poses the question: how can clickstream behavioural patterns, as identified by machine learning algorithms, be visualised in order to identify website (dis)satisfaction areas on an e-commerce website? To assess the research question, this study proposes an application assessment framework, identifies relevant explainable machine learning methodologies and the most important aspects of clickstream data, as related to website satisfaction. To link clickstream behaviour to website satisfaction levels, this study argues that sessions without (or low) purchase intent should be eliminated before assessing the data. By fitting several gradient boosted tree models onto data from a major European e-commerce fashion store, this study establishes that static aspects of clickstream data are the most important predictors of website satisfaction levels and demonstrates what an application that identifies website (dis)satisfaction patterns might look like. Combining static data with the temporal and sequential aspects of clickstream data renders the best predictive performance.*

## 1. Introduction

As the world becomes increasingly digitalised, user experience design is becoming evermore relevant (Chandramohan & Ravindran, 2018; Filipowska et al., 2019; Raphaeli et al., 2017). From a web design perspective, user experience designers, web developers and marketing professionals (collectively referred to as UX professionals in this study) concern themselves with creating web experiences that enable users to achieve various tasks with ease, while making a pleasant experience out of it (Cai et al., 2018; Narang et al., 2017). Within e-commerce, making a successful purchase is often the central task (Cai et al., 2018), and successful task achievement is often referred to as a conversion (Gudigantala et al., 2016).

Efforts to increase conversion are on the rise, yet less than a third of the implementations produce results (Econsultancy & RedEye, 2018). A report conducted by the *CXL Institute* and *AB Tasty,* for example, revealed that better *processes* and better *optimizers* are two of the top three challenges UX professionals face; increasing conversion rates ranked seventh (Gleason, 2019).

To improve conversion rates, UX professionals gather and analyse users' feedback in order to inform web adjustments, often in an iterative manner (Narang et al., 2017; Padidem & Nalini, 2017). Clickstream behaviour pattern visualisation is widely used by practitioners to gather user behavioural feedback (Liu et al., 2017). To this end, the need for visualising clickstream behaviour has been identified more than a decade ago (see Kohavi, Zheng, Lavrač, Motoda, & Fawcett, 2004, for example) and commercial clickstream visualisation tools are widely available (Liu et al., 2017).

However, clickstream behaviour is often complex and simple visualisations fail to capture and illustrate the complexities between clickstream behavioural patterns and conversion (Chandramohan & Ravindran, 2018; Filipowska et al., 2019; Liu et al., 2017). Raphaeli et al. (2017), for instance, identify that static (i.e. the number of times a particular page was viewed), temporal (i.e. session duration, or page dwell time) and sequential (i.e. frequent sequences of events in a session) features play a role in explaining online behaviour. In spite of this, visualising the complex relationships between conversion rates and clickstream behaviour has not been addressed sufficiently (Filipowska et al., 2019; Liu et al., 2017).

Meanwhile, machine learning (ML) has been successfully applied to identify clickstream behaviour patterns in order to predict conversion (Bigon et al., 2019). Yet, while machine learning is well suited to visualise clickstream patterns (Lundberg et al., 2020), less effort has been dedicated to visualising the complex relationships between clickstream behaviour and conversion (see Filipowska, Kaluzny, & Skrzypek, 2019 for an overview). At the same time, UX professionals identify the need of integrating machine learning into their practices (Saket et al., 2018), but many practitioners do not know how best to leverage its potential (Dove et al., 2017).

In addition to the challenges associated with visualising (non-)converting clickstream behavioural patterns, the link between clickstream behaviour and website (dis)satisfaction needs to be established. This is because people visit e-commerce websites without necessarily intending to make a purchase (Raphaeli et al., 2017). Visitors might be visiting a website to perform a price comparison, or just 'window shop', for example. In such cases, visualising the (non-)converting behavioural patterns would illustrate behavioural differences between, say, visits with a 'window shopping' intent and purchase intent visits. Thus, without establishing the link between clickstream behaviour and website (dis)satisfaction, such visualisations are of limited use to UX professionals because they provide noisy (if any) information with regards to website satisfaction levels.

These challenges leave a wide research gap open, allowing future studies to focus on how explainable machine learning (XAI) techniques can be employed to identify and visualise the complex relationships between clickstream behaviour and conversion rate. From a practitioner's point of view, such tools could enable UX professionals to extract valuable feedback from these complex relationships, and, therefore, increase conversion.

This project aims at filling this gap by assessing how machine learning can be used to identify and visualise clickstream data patterns, such that UX professionals can gain insights into the complex relationships between clickstream behaviour and users' web satisfaction levels through visualisations, as assessed from an e-commerce perspective. To inform this assessment, existing literature accompanied by insights drawn from a major European e-commerce fashion store's clickstream data are used. The main research question this study addresses is:

> *RQ: How can clickstream behavioural patterns, as identified by machine learning algorithms, be visualised in order to identify website (dis)satisfaction areas on an e-commerce website?*

To facilitate pattern contrasting, the problem is phrased as a binary classification (*purchased* or *not purchased*) machine learning problem. This question, in turn, is broken down into three sub-research questions (SRQ's):

> *SRQ1: Which aspects of clickstream data can predict conversion better? That is, do the static, temporal or sequential aspects of clickstream data help predict conversion? If so, which of the types renders the best performance? Does using specific combinations of static, temporal and sequential data outperform a*

*model trained on a single clickstream data type? Does using all three data types render better performance?*

*SRQ2: What criteria should be used to assess how good an application aimed at illustrating the (non-)converting patters and, by extension, website satisfaction is? Specifically, what information should visualisations convey in order to illustrate what the differences between the converting and non-converting patterns are? And, under what circumstances does contrasting these patterns illustrate website (dis)satisfaction?*

*SRQ3: What XAI methodology, that adheres to the criteria identified in SRQ3 could be applied in order to allow UX professionals to discover possible website (dis)satisfaction areas on an e-commerce website?*

To answer the first sub-question, static, temporal, and sequential features, as well as combinations thereof, are modelled using an XGBoost algorithm and then, their predictive performance is assessed and contrasted. The second and third sub-questions are answered in two parts. First, insights drawn from the fields of social sciences, human-computer interaction (HCI), and explainable machine learning (XAI) inform an assessment framework, denoting what important considerations need to be made in preparing and modelling the data and how visualisations should be provided. Second, visualisations illustrating the (non-)converting patterns, as learned by a trained ML model, on the aforementioned online clickstream data, are produced and assessed against the proposed framework. This study concludes that, although combining all aspects of clickstream data renders the best predictive performance, static features are the most important features in predicting conversion and, by extension, website satisfaction levels. To link online clickstream behavioural patterns to website satisfaction, this study argues that sessions where purchase intent is low or non-existent should be removed before assessing the data.

## 2.Related Work

This section is made up of five subsections. The first two subsections discuss how clickstream behaviour could be related to website (dis)satisfaction, the clickstream feature types that could be used to identify (dis)satisfaction and how clickstream data should be pre-processed.

Next, subsections 2.3 and 2.4 focus on identifying relevant explainable machine learning methodologies. In Subsection 2.3 insights form the social sciences and the human-computer interaction (HCI) fields of study are linked to the field of explainable machine learning (XAI). Then, in Subsection 2.4, existing XAI methodologies are discussed in light of the theoretical framework identified in Subsection 2.3. Based on existing literature, an assessment framework is developed next, thus informing SRQ2. Lastly, in Subsection 2.5, existing XAI applications aimed at illustrating clickstream behaviour are reviewed in light of this assessment framework.

### 2.1 Intention to purchase and clickstream data

Collecting web-satisfaction data from approximately twenty thousand people and conversion rates from 85 leading e-commerce stores in the United States, Gudigantala et al. (2016) find that, controlling for average item cost, website satisfaction has a statistically and economically significant impact on conversion rate. The effect remains positive even

when intention to purchase was present at the beginning of the visit. Several other studies found that a website's quality impacts users' intention to use the website (see Hartono & Holsapple, 2019 for an in-depth discussion). This indicates that website satisfaction impacts conversion rates.

However, there are a multitude of reasons for visiting an e-commerce platform, ranging from 'window shopping', to price comparison and purchasing (Raphaeli et al., 2017). And, since machine learning, essentially, picks up the underlying patterns in data (Hastie et al., 2009), it would be difficult to attribute the differences between converting and non-converting patterns to website satisfaction. For example, sessions where users were 'window shopping' are less likely to lead to a conversion than sessions where users intended to make a purchase, and they might have a different clickstream behavioural pattern. Claiming that the clickstream behavioural differences between these two types of sessions are linked to website satisfaction would be misleading. Thus, assessing sessions where an intention to purchase is present provides an acceptable trade-off between losing information related to website satisfaction and removing the noise accompanied by visits where no purchase intent was present.

In spite of the necessity to identify it, understanding intention to purchase is multifaceted, and several long and short term variables are likely to impact a visitor's intent to make a purchase (Lo et al., 2016). It is thus difficult to distinguish sessions that had an intention to purchase to begin with, from those that did not.

Some have been successful at identifying purchase intent though. Zhao et al. (2020) illustrate, by using six months' worth of e-commerce data from etsy.com (a craft item e-commerce platform acting as a marketplace), that adding an item to basket "exhibits a much higher buying intention", compared to sessions where no items were added to basket (X. Zhao et al., 2020, p. 456). This indicates that using the add to basket event as a proxy for intention to purchase filters out, at least to some extent, sessions where there was no intention to purchase. Thus, by filtering out sessions where no items were added to basket, the differences between converting and non-converting patterns are more likely to be attributed to website satisfaction. This is further illustrated with the aid of a diagram below (Figure 1).

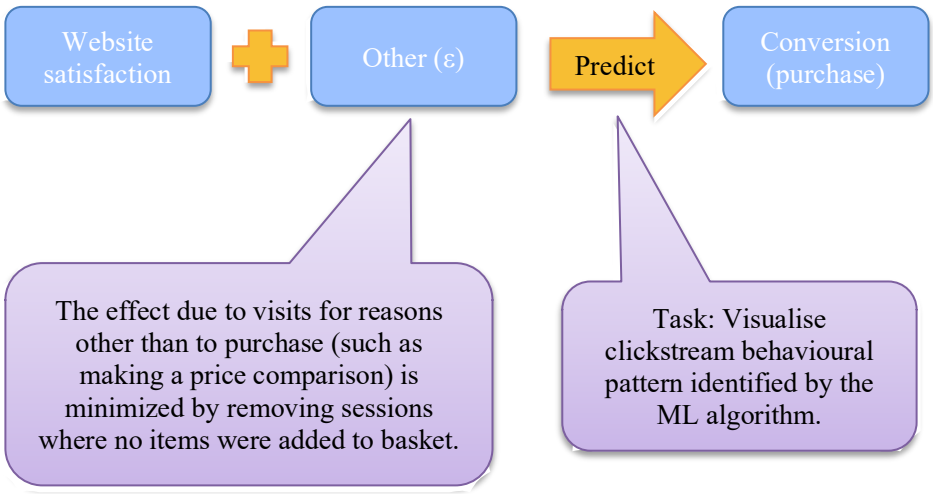


*Figure 1. The figure illustrates that besides website satisfaction, other factors influence the outcome of a conversion. By treating the add to basket event as a proxy for purchase intent and removing sessions for which no items were added to basket, the ML visualisations are more likely to reflect behavioural patterns related to website satisfaction.*

### 2.2 Pre-processing and features to be extracted

In an effort to identify which clickstream behavioural variables correlate with purchases, Raphaeli et al. (2017) review existing literature and identify the number of session events, session duration, count of pages viewed, the average time spent on a page, and click event paths to be correlated to purchasing behaviour. This suggests that click path sequences, time spent on events and the nature of the event are likely to help predict conversion. UX professionals could, therefore, gain insights by analysing differences between converting and non-converting clickstream behavioural patterns by assessing all the above-mentioned variables.

Applications aimed at improving conversion often represent website sessions as full-length sequences of clickstream events (such as the sequence *viewed page, viewed product, added to basket*) (see Bigon et al., 2019, for example). While these representations might render better predictive performance, event-rich website visits can be difficult to visualise (Chandramohan & Ravindran, 2018). Therefore, this study proposes limiting the sequential data to a small number of sequences (n-grams).

### 2.3 Explainable ML

Miller (2019) dubs the XAI field of study as Human-Agent Interaction (HAI) and argues that it lies at the intersection of Human-Computer Interaction (HCI), Artificial Intelligence (AI), and the Social Sciences fields of study, as illustrated in Figure 2 below.

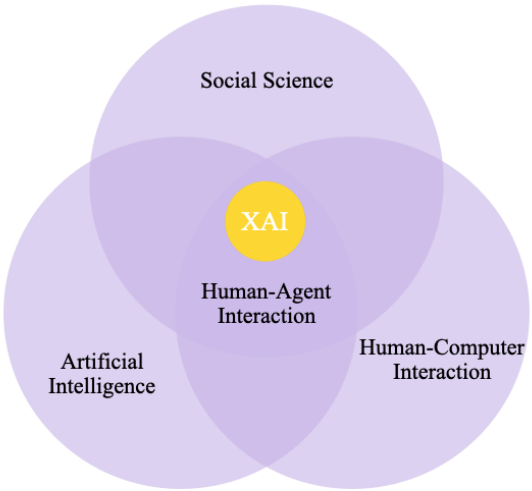


*Figure 2. The scope of explainable artificial intelligence (XAI). Based on* (Miller, 2019).

Miller's (2019) study provides a broad taxonomy of what good explainable artificial intelligence (XAI) is and argues that most XAI research focused on the AI and HCI fields of study, in spite of the fact that, what constitutes a good explanation, is a well-researched and mature field of study in the Social Sciences. Miller's (2019) taxonomy of XAI is used to inform this project.

### 2.3.1 Human-Agent Interaction and Social Sciences

In an extensive literature review, Miller (2019) draws insights from the Social Sciences to identify four main aspects which characterise best explanation practices.

First, Miller (2019, p. 3) argues that explanations should be *contrastive*, because "people do not ask why event P happened, but rather why event P happened instead of some event Q". For UX professionals, this implies that an explanation should allow one to determine *how* a feature affects the likelihood of a session ending in a conversion.

The second finding is that people tend to *select the aspects of an explanation in a biased manner*. That is, while people do not typically expect explanations to provide the complete picture of an event, they are "adept" at picking a subset of explanations which serve them best, rather than the explanations which are most likely to paint an accurate picture of the event (Miller, 2019, p. 3). For UX professionals this implies that an explanation should clarify which features are the most important for predicting conversion.

The third finding is that, in spite of their practical importance, for people, *probabilities are less important than causes* (Miller, 2019). That is, explaining statistical relationships is less likely to be as effective at successfully conveying an explanation as assigning a cause. In order to successfully communicate explanations to UX professionals, the features most 'responsible' for conversion need be identified. This can be achieved by illustrating the feature importance (which also satisfies the second finding's requirements).

The fourth finding, "*explanations are social*", states that good explanations should be presented such that they illustrate both the explainer and the explainee's beliefs, or reference points, much like a conversation does (Miller, 2019, p. 3). This last dimension of explanations deals with the narrative aspects of explanation and is deemed outside the scope of this project.

Taken together, Miller's (2019) four findings indicate that, in order to allow UX professionals to assign 'causes' while limiting bias in feature selection, the XAI visualisations should identify the most important features for predicting an outcome. Then, in order to allow for contrastive assessments, the XAI visualisations should illustrate how the features impact a prediction made (feature effects).

### 2.3.2 Human-Agent Interaction, Human Computer Interaction, and XAI

The human-computer interaction field of study is broad and draws from several other fields of study, from psychology and cognitive sciences, to ergonomics and computer science (Dix, 2017). HCI, however, is as much an applied discipline as it is theoretical and distinguishing between the two would be difficult, possibly misleading (Dix, 2017), and goes beyond the scope of this study.

From a practitioner's perspective, Wang et al. (2019) draw on concepts from theoretical HCI models and current XAI methodologies, in an effort to develop an XAI practitioner-ready framework. Their framework follows a three-step approach – first develop an understanding of user reasoning and consider user biases, then identify the most suitable XAI representations and, lastly, develop and deploy the XAI representations. Figure 3 (in Appendix A), illustrates the links between HCI theoretical models and appropriate XAI representation methodologies.

Regarding the first step, Miller's (2019) findings help develop an understanding of user reasoning, and possible biases. In order for to allow UX professionals to assign 'causes' while limiting bias in feature selection, the XAI visualisations should therefore identify the most important features for predicting an outcome. Then, in order to allow for contrastive assessments, the XAI visualisations should illustrate how the features impact a prediction made (feature effects).

Regarding suitable XAI representations, as the second step, Wang et al. (2019) framework points towards partial dependence plots (PDP), tornado plots or, for image-recognition tasks, saliency heatmaps, because these representation would allow for

contrastive and attributive interpretations. In order to illustrate the features most responsible for a prediction, and therefore enhance attribution, feature importance could be plotted. The third step, deployment, is outside the scope of this study.

Summed up as an evaluation framework, XAI applications should be contrastive and attributive, while the clickstream features should contain temporal (such as pageview dwell time), sequential (i.e. frequent sequences of events in a session), and static information (such as number of visits to a specific page).

Lastly, because the aim of this study is to develop an XAI methodology which allows UX professionals to identify possible website dissatisfaction areas on an e-commerce platform, it is argued that add to basket events should be used as a proxy for purchase intent, and sessions without an add to basket event should be removed from the analysis. This is because, since users visit an e-commerce platform for multiple reasons, contrasting behavioural patterns is likely to reveal user intent patterns (such as 'window shopping' versus intention to purchase) rather than patterns related to web satisfaction. The proposed framework is illustrated in Figure 4 below.

| ***Criteria*** | ***Brief XAI application assessment description*** |
|---|---|
| *Purchase intent* | In order to be able to attribute feature effects to web satisfaction, clickstream data without a purchase intent (task-achievement intent) should be removed. |
| *Contrastable* | Explanations (visualisations) should allow UX professionals to contrast converting patterns against non-converting ones. |
| *Attributive (which)* | Explanations (visualisations) should allow UX professionals to observe which are the most important features in determining the outcome of a prediction. |
| *Attributive (how)* | Explanations (visualisations) should allow UX professionals to observe how a feature determines the prediction outcome. That is, how specific values of a feature affect the prediction outcome. |
| *Features* | Temporal (such as session duration), sequential (such as common event sequences) and static features (such as product detail views, or count of items added to basket) should be considered in the analysis. |

*Figure 4. Assessment framework. The table illustrates the framework used for evaluating how 'good' an explanation is at allowing UX professionals to identify possible web (dis)satisfaction areas on an e-commerce platform.*

By reviewing literature aimed at (1) identifying how clickstream data should be pre-processed and (2) assessing how suitable an XAI methodology is for contrasting converting and non-converting clickstream behavioural patterns, a remedy to the sub questions addressed in this study has been provided. Therefore, existing applications are assessed against this proposed framework.

### 2.4 Explainable AI (XAI) methodologies and machine learning algorithms

#### 2.4.1 Explainable AI methodologies review

This subsection provides a succinct outline of the methodologies used to visualise, or otherwise explain, how supervised machine learning models make use of the input data to make a prediction. Tangencies between XAI methodologies and clickstream data

visualisation are briefly discussed too. Lastly, the models and methodologies applied in this study are revealed.

While popular in academia and amongst practitioners, the idea that there is a trade-off between model accuracy and interpretability is debatable (Rudin, 2019). Rudin (2019) makes a compelling argument that, for most data science applications, there is little evidence, if any, that black box models, such as neural network or random forest models, render better performance than more interpretable models, such as a logistic regression model.

However, Rudin (2019, p. 1) mentions that domain-specific expertise for feature-engineering purposes is needed in order to develop interpretable models, while emphasising that interpretable models are important for "high-stake" decisions, such as explaining a bank-loan granting decision. Although using less complex, interpretable, models would generate more accurate representations of clickstream behaviour and could be rewarding to explore, due to lack of access to domain expertise (UX professionals) and development time, this study's analysis is limited to black box models.

For black box models, several model agnostic methodologies have been developed to aid interpretability via visualisations (see Lundberg & Lee, 2017 or Molnar, 2020 for an in-depth overview of some of the model agnistic methodologies). Essentially, model-agnostic methodologies produce explanations after a model has been trained (post-hoc) by learning and illustrating the features' effect form the predictions a black box model makes (Ribeiro et al., 2016). Such methodologies have the advantage of flexibility, thus allowing the XAI developer to switch between models with ease; their main disadvantage is that the explanation is a representation of the black box model, which can be misleading (Ribeiro et al., 2016).

In order to explain which features are important, this study proposes illustrating a model's variable importance plot, using Fisher et al. (2019) methodology. Fisher et al. (2019) methodology compares the original model's prediction error with the prediction errors produced by models where each of the features were shuffled, one at a time. The underlying idea behind this methodology is that, if a feature is not very important, then the errors will not change much when shuffling a feature. Applying this methodology to identify how important a feature is, satisfies the which-attribute aspect of the proposed framework (Figure 4).

Wang et al. (2019), as discussed in the previous subsection, proposed using partial dependence plots (PDP) in order to satisfy the how-attribute of the proposed framework. PD plots, first proposed by Friedman (2001), illustrate how the outcome of a prediction depends on the values of a specific feature. By altering the values of the feature to be plotted, while keeping the rest of the features constant, the PD methodology computes the average predictions of a model in order to establish feature importance. Note that, because the feature of interest's values are altered while others remain constant, if the altered feature is correlated with any of the other features, unrealistic combinations of the feature in question are introduced. For example, it might be unlikely that a two-meter-tall person weighs less than 50kg (there is a correlation between height and weight), while the PD algorithm would introduce such an unrealistic value and force the model to make a prediction (Molnar, 2020). Thus, PDP methodology implicitly assumes that there is no (or little) correlation between the explanatory variables, which is an unrealistic expectation.

As an alternative, the accumulated local effects (ALE) plot of a feature, a methodology proposed by Apley and Zhu (2016), provides major improvements over the PD methodology. ALE's use the conditional distribution of a feature in order to avoid introducing unrealistic values. The algorithm works by dividing a feature's space is into several small intervals and replacing the original feature's value with small variations of the value, as follows.

$$\hat{\tilde{f}}_{j,ALE}(x) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{i:x_j^{(i)} \epsilon N_j(k)} [f\left(z_{k,j}, x_{\setminus j}^{(i)}\right) - f(z_{k-1,j}, x_{\setminus j}^{(i)})] \quad (1)$$

In equation (1) the uncentered effect is calculated (Molnar, 2020). Reading the equation from the right, $z_{k,j}$ represents instance j of interval k. This is the variation of the feature's original value. The difference in prediction outcomes between instance $z_{k,j}$ and $z_{k-1,j}$ is then calculated, with $x_{\setminus j}^{(i)}$ as input for the remainder of features associated with the original instance, *i*. Next, differences are summed up for all predictions in the neighbourhood $N_j k$ and divided by $n_j(k)$, the number of instances *j* in interval *k*, in order to compute the average effect. This average effect is then accumulated across the intervals *k*.

In other words, variations close to the feature's original values are inserted into the model in order to generate predictions. The *differences* in outcome predictions made by a model are averaged as the feature's importance. After the procedure from equation (1) is completed, the effect is centred to have a mean of zero.

ALE's illustrate a feature's local effect (for a specific feature value) (Apley & Zhu, 2016). Accumulating the local effects of a feature, allows one to illustrate the 'global', overall effect of the feature on the prediction outcome (Apley & Zhu, 2016).

Without impacting on interpretability, ALE's have two main advantages over PDP's. First, and in contrast to PDP's, by computing the differences within these intervals, the ALE methodology avoids including the effect other correlated features might have on the prediction outcome (Molnar, 2020). Second, they are computationally cheaper than PDP's (Molnar, 2020), making them a more suitable choice for application embedment.

ALE's main disadvantage is that, depending on the number of intervals chosen, the ALE plot can either hide the true complexity of a model (when a small number of intervals is chosen) or generate a 'shaky' plot, making it difficult to interpret when a large number of intervals is chosen (Molnar, 2020).

### 2.4.2 Application methodology

Altogether, the ALE methodology is robust to correlated features, and because the ADL strikes a balance between ease of interpretation and true representation (see Molnar, 2020 for an in-depth comparison of variable effect plotting methodologies), this study proposes illustrating the how-attribute via a ALE feature effect plots.

In spite of the wide variety of methodologies available for XAI, the largest body of literature dealing with the complexities of clickstream data visualisation is limited to pattern mining (see Filipowska et al., 2019 for a brief overview), and Markov-based models (see Brainerd & Blue, 2001; Frhan, 2017; Kateja et al., 2014, for example). For predicting conversion form clickstream behaviour, however, a wider range of models, including Markov-based models, neural networks (see Koehn et al., 2020, for a brief review), or tree-based models (Mokryn et al., 2019; Sheil & Rana, 2018), or even combinations of models (Mokryn et al., 2019), have been implemented and rendered comparative predictive performance. Lastly, Koehn et al. (2020, p. 1) note that recurrent neural networks (RNN's) and "conventional classifiers" identify "different patterns in clickstream data" and recommend that different classes of machine learning algorithms should be used together, in order to capture them.

This study uses the XGBoost tree-based model to illustrate the methodology. This choice is justified by the fact that Mokryn et al. (2019), find the algorithm to render good performance when compared to other best-in-class algorithms, on similarly pre-processed clickstream data. This methodology could be applied to other classes of ML algorithms, or even combinations thereof, as suggested by Koehn et al. (2020). However, since a

performance comparison between classes of machine learning algorithms is beyond the scope of this project, and because of implementation time constraints, other classes of ML algorithms, or combinations thereof, will not be considered in this study.

### 2.5 Existing applications

Filipowska et al. (2019) propose a process mining system to assess clickstream data, whereby desirable navigation sequences are correlated with actual navigation sequences, as identified via a pattern mining algorithm. Filipowska et al. (2019) first use the data to identify clickstream patterns, and then they relate it to the desired clickstream paths and task-achievement, by means of a correlation matrix, as their explanatory visualisation. Such a system would allow UX professionals to contrast the navigational patterns intended for task-achievement against the actual clickstream patterns which most frequently lead to conversion.

Filipowska et al. (2019) incorporates sequential features but no temporal or static features are included in their analysis. While it could be argued that the correlation matrix between desirable and actual sequences allows one to contrast converting patterns against non-converting, and that such contrasts could allow UX professionals to attribute conversion (or lack thereof) to specific sequences, Filipowska et al. (2019) study does not illustrate how specific features influence conversion.

Similar to Filipowska et al. (2019) study, Raphaeli et al. (2017) use the event paths as input for a sequential pattern mining application, in an effort to compare purchasing behaviour between mobile and PC e-commerce platforms. The paths are captured differently, however. First, the event paths are captured by classifying events as *up*, *down* or *same*, where up is a page which was visited for the first time during the session, down if a previous page was visited next in the sequence, and same if the same page is visited again (as the result of, say, refreshing the page). Then, the sequences of up, down and same are summed up further as either *fingers* (down-up-down), *upstairs* (sequential up, with the possibility of same occurring in-between), *downstairs* (sequential down, with the possibility of same occurring in-between), or *mountain* (a combination of upstairs and downstairs). Figure 5 illustrates these sequences.

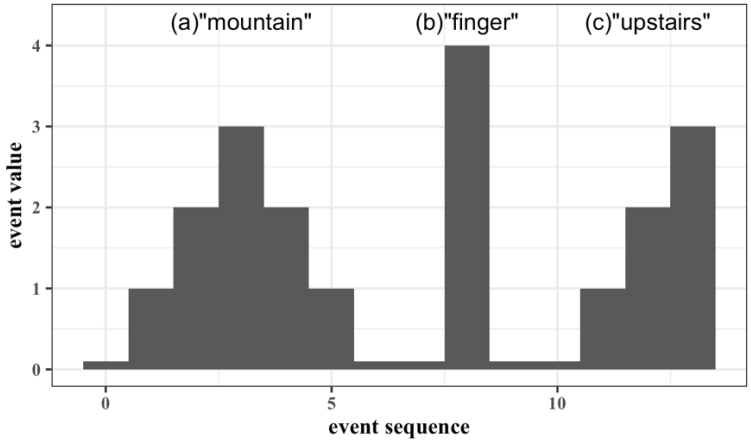


*Figure 5. Mock session illustrating the Automatic Pattern Discovery (APD) sequence classification technique. The figure illustrates a fictive sequence of (a) a mountain, (b) a finger, and (c) an upstairs sequence. Based on* Raphaeli et al. (2017).

After encoding the clickstream data into the above-mentioned sequences, Raphaeli et al. (2017) spilt the data into buying and non-buying sessions and apply a pattern mining algorithm to identify the most common patterns. Contrasting the most common buying and non-buying patterns could allow UX professionals to identify areas which need attention on an e-commerce platform.

In contrast to Filipowska et al. (2019) study, Raphaeli et al. (2017) make use of the temporal aspects of clickstream data in order to identify differences in temporal behaviour between buying and non-buying sessions. The temporal analysis is limited to contrasting the aggregate statistical characteristics between converting and non-converting sessions, however. Specifically, Raphaeli et al. (2017) conduct t-tests to check for statistically significant differences in session duration and average page duration between converting and non-converting sessions. While these temporal behaviour insights could be insightful to UX professionals, they fail to capture more complex relationships between temporal behaviour and the likelihood that a purchase is going to be made during a session.

While Raphaeli et al. (2017) study is contrastable and it captures sequential and temporal aspects of clickstream behaviour, it does not make use of any stationary data, and attribution is limited to *which* features are linked to conversion, without explaining *how*.

Several other studies have applied pattern mining algorithms to assess clickstream data, including Dias and Ferreira (2017), Liu et al. (2017, 2016), and Su and Chen (2014), with varying degrees of focus on pattern visualisation. However, they all fail to capture the temporal aspects of clickstream behaviour and the *how* aspects of attribution.

Another popular way of visualising clickstream data is by means of Markov chains.

Brainerd and Blue (2001) make use of Markov chains to visualise clickstream data. Their visualisation (observed in Figure 6, below) shows the overall flow of clickstream events. By adjusting the directional arrow's thickness to represent the proportion of sessions which followed a particular subsequent event, Brainerd and Blue (2001) allow UX professionals to contrast converting and non-converting patterns. It also allows for what-attribution and the Markov chain implicitly captures static information.

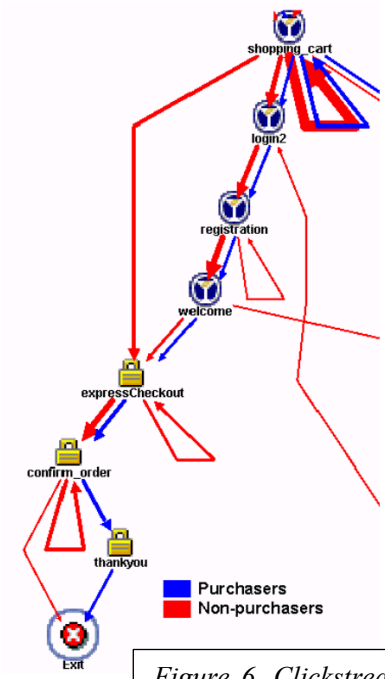


*Figure 6. Clickstream event flow, starting from the shopping cart, illustrating the differences between buying and non-buying sessions. Reprinted from* (Brainerd & Blue, 2001, p. 4).

The main limitations of Brainerd and Blue (2001) study are threefold. First, their methodology does not include any temporal aspects of a session. Second, much like the pattern mining applications, the how aspects of attribution are not captured by their application either. Third, because sessions tend to be long (some more than 200 events), and because of the numerous combinations of click paths that take place on an e-commerce platform, an overall click path visualisation would be large and difficult to visualise.

More recent methodologies, such as VizClick (Kateja et al., 2014) or *WebClickViz* (Frhan, 2017), made progress with regards to the visual representation of lengthy, complex patterns, making Markov chain-based applications easier to visualise. However, the how aspects of attribution and the temporal aspects of a session have not been included in these recent methodologies either.

Neural networks, and other black box models can also be used for visualisation (see Krause et al., 2016, for example). Though their intent is to advance conversion rate prediction performance, Chandramohan and Ravindran (2018) study captures sequential, static and temporal elements of clickstream data and produce visualisations which could be valuable to UX professionals. After training a neural network model, they present a heatmap illustrating which of the features are important for predicting conversion, for each event in a clickstream sequence, as illustrated in Figure 7, below.

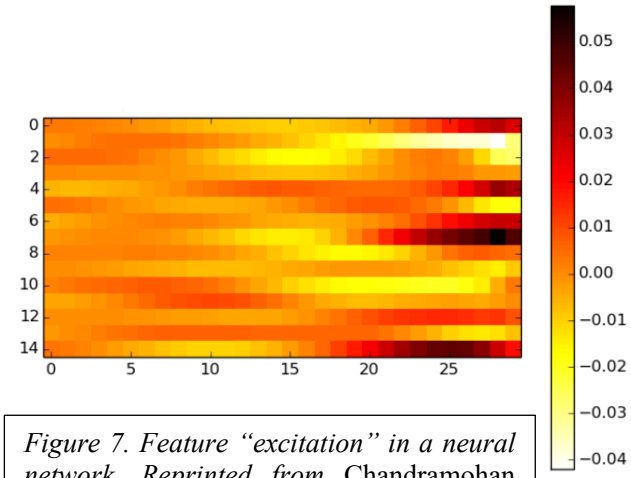


*Figure 7. Feature "excitation" in a neural network. Reprinted from* Chandramohan and Ravindran (2018, p. 9).

The visualisation in Figure 7 illustrates a set of 14 categorical features and how they become more (darker) or less (lighter) important in predicting a converting clickstream (a purchase). Although Chandramohan and Ravindran (2018) admit to having difficulty interpreting visualisation of numerical features, this visualisation provides contrastable and attributive insights. The sessions are capped at 30 events, however, and allowing for lengthier sessions could make the interpretation of such visualisations more cumbersome.

Other existing methodologies, either illustrate how the proportion of clicks form one page to another flow (i.e. *product detail view – add to basket*) (J. Zhao et al., 2015), which fail to capture complex relationships; apply lengthy, in-depth analyses which are too broad in scope for an informative XAI tool (G. Wang et al., 2016; Wei et al., 2012); or enrich their clickstream data with additional, personal information (Conglei Shi et al., 2015).

Overall, the literature on visualising clickstream behaviour has three main limitations. First, none of the applications intended at assessing website usability deal with visit intent. If intention to purchase is not, at least to some extent, present at the beginning

of a session, the patterns identified, however well explained, would be misleading. This is because the patterns identified are likely to describe sessions where users were not concerned with purchasing at the time of visit. Second, most applications do not focus on all three behavioural aspects of clickstream behaviour. While most focus on the sequential aspects, fewer focus on the temporal or static aspects of a session, and none focused on all three. Third, most studies provide contrastable explanations and focus on the which aspects of attribution, neglecting the how aspects of attribution.

Altogether, these limitations leave a wide gap open for research on clickstream data visualisation methodologies which encapsulate the complexities of static, sequential and temporal aspects of online behaviour, as they relate to conversion.

## 3. Experimental Setup

The entire experiment was conducted in the *R* programming language (version 3.6.3), with *RStudio* (version 1.2.5042) as the interactive development environment (IDE). The data pre-processing was completed using the *tidyverse library* (*Tidyverse*, n.d.). The algorithmic modelling was implemented using the *tidymodels library* (*Tidymodels*, n.d.). Other packages used peripherally will be mentioned in the methodology subsection.

This rest of the section is made up of two subsections. The first describes the data, its provenance, characteristics, and how it has been pre-processed for this project. The second subsection provides a detailed description of this study's methodology.

### 3.1 Data

The clickstream data is extracted from a major European fashion e-commerce store, as observed over the period starting June 6$^{th}$, 2018 until July 7$^{th}$, 2018 (a month). The data has been made available by Tooso, an AI-based retail solutions start-up, and it is originally described and used by Bigon et al. (2019).

The raw dataset is made up of approximatively 5.43 million clickstream events observed over 443,663 web visits (~12.2 events per visit). A clickstream event can represent a page visit, a product detail view, that a product was added to basket, that a product was removed from basket, that a purchase was made, or an undetermined click action. Each of the events is accompanied by a timestamp, which allows for a session to be observed in the sequence the events took place. Lastly, a hashed product code and web address are also available but these features are disregarded in this analysis.

To isolate sessions for which an intention to purchase was present, at least in part, all sessions for which no products were added to the basket are removed. The remaining clickstream events represent 40,0076 sessions – that is 9.03% of the all the sessions observed over the period. All events which occurred after a purchase event are removed from the remainder of the sessions. That's about 200,000 events.

Then, after removing all sessions with less than 10 events or more than 200, in line with Bigon et al. (2019), a total of 33,386 sessions are left, with a mean of 44.71 events per session. The events per session histogram below (Figure 8) illustrates a positively skewed distribution, with most sessions made up of less than 50 clickstream events. Switching between events took an average of 46.3 seconds, with an upper quartile of 43 seconds (also positively skewed, as illustrated in Figure 9, Appendix B).

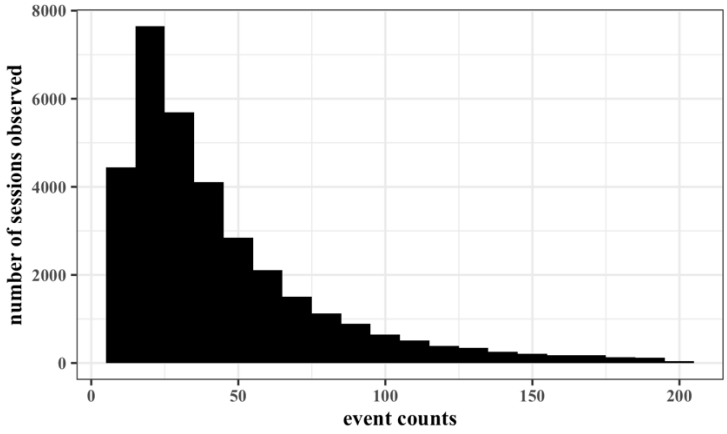


*Figure 8. Histogram illustrating the events per session.*

Next, the clickstream data is aggregated at session level (sessionalised), such that each session is represented by ten variables, six static and four temporal. Five of the static features illustrate the number of times a visitor had a pageview, a product detail view, an undetermined click, and an item added or removed form basket. The sixth static feature is the count of total events per session. The temporal features represent the average time, in seconds, a user spent on performing specific events. Three are computed as the average time per session spent on a pageview, a product detail pageview, or an add to basket event. The fourth temporal feature represents the total session duration. Because by the time a user removes the item from the basket, the website has already been experienced, and since most sessions do not contain an undetermined click event, the click dwell time and remove from basket dwell time features are left out of the analysis.

Of the 33,386 observed sessions, 597 of them are missing values for one or more of the features. This is could be due to various reasons regarding the way sessions were logged. The observations missing the time taken to add to basket, for example, occur because the first event taking place in a session is the add to basket event. These sessions are dropped from the summarised dataset, such that 32,789 sessions are now observed.

The sessionalised dataset is enriched with sequential behavioural features next. To capture the sequences, the 2 and 3 n-grams of events are computed using the ngram package (R-project.org, 2017). Then, sequences occurring at least 5% of the time are identified, for each of the n-grams calculated. The procedure identifies 11 n-grams - six digram sequences and five trigram sequences, as illustrated in Figure 10, Appendix B. Finally, the dataset is enriched with the 11 additional features, each representing the number of times such a sequence occurred.

It should be noted that, since purchase events are the target variable, all purchase events were removed before summarising the data, in order to 'blind' the model any possible information regarding conversion outcome. The summarised dataset is enriched with the binarized target variable, *purchased,* afterwards.

The pre-processed dataset is made up of 21 explanatory variables, with a total of 32,789 sessions observed. Of the 21 features, 11 are sequential, 6 are static and 4 temporal. The conversion rate is 18.76% (6,512 of the sessions had a purchase). Figure 11 in Appendix B presents the dataset's summary statistics.

### 3.2 Method / Models

The focus of this analysis is to visualise the clickstream behavioural patterns leading to a conversion and contrast them against the patterns which do not lead to a conversion, by illustrating how the features impact the prediction outcome.

The proposed methodology is made up of three parts. First, a preparatory analysis is performed, in order to 'get to know' the data, its characteristics, and conduct the preliminary preparations before modelling. Then, in order to facilitate fine-grained hyperparameter tuning, a two-stage modelling process is proposed. Third, the explanatory visualisations are produced. The remainder of this subsection describes these parts in detail.

#### 3.2.1 Preparatory analysis

In this project a session's purchase outcome is the target variable. Hence, the problem is treated as a binary classification problem, where the outcome of a session is either a purchase or not.

Before continuing with the modelling, the sessionalised dataset is split randomly in two stratified subsets, such that 70% of the data (22,953 observed sessions) are used for training the algorithms (henceforth referred to as the training dataset) and the remaining 30% of the data (9,836 observed sessions) is used to assess how well the trained model is performing on 'unseen' (testing) dataset. The testing subset is therefore excluded from all but the final performance assessment part of the analysis process.

A Pearson correlation matrix is then produced as illustrated in Figure 12, Appendix B. Because some pairs of variables are nearly collinear (correlation > 0.9), highly correlated variables are removed at random, for each pair, such that only one of the highly correlated features remains. Note that, with the exception of the add to basket feature (which is among the removed features), only features of the same type are highly correlated. More specifically, sequential features tend to be highly correlated with other sequential features.

The sessionalised dataset is now made up of 15 features (6 removed), five static, four temporal, and six sequential. The removed features are clearly marked in Figure 11, Appendix B.

An initial visual analysis, contrasting the 15 explanatory variables' converting and non-converting probability distribution functions (PDF), is conducted before modelling. Three figures are produced to this end and are illustrated in Appendix C. Figure 13 illustrates the 5 static features' PDFs, Figure 14 illustrates the 4 temporal features' PDFS, and Figure 15 illustrating the 6 sequential features' PDFs.

Finally, the training data is downsampled, so that an equal amount of purchasing and non-purchasing sessions is observed in the training subset (12,304 sessions observed after downsampling).

#### 3.2.2 Modelling

The modelling and evaluation process is carried out in the tidymodels library, which provides a collection of R packages, altogether facilitating a beginning-to-end machine learning modelling framework (*Tidymodels*, n.d.).

A boosted tree algorithm is an ensemble algorithm which works by 'boosting' the observations for which a previously fitted decision tree predicted erroneously (Hastie et al., 2006). A boosted tree algorithm works stagewise, initiating with a decision tree on the data. Then, it computes the residuals (erroneous predictions), and selects them in order to fit another tree (hence boosting them). Each consequent tree updates the previous one, to form ensemble of trees, until some stop criteria is reached. The model is implemented in R via the XGBoost package (Chen et al., 2020), which is a considerably faster implementation of the boosted tree algorithm methodology (Chen & Guestrin, 2016).

The aim of this application is to train a model which 'learns' how to correctly identify the positive class – a purchase being made. Imposing a threshold to make a prediction should involve business consideration. Because it illustrates how the model performs at various thresholds, this project proposes the receiver operating characteristic area under the curve (AUC) as the key performance metric.

#### 3.2.2.1 Hyperparameter tuning procedure and model fitting methodology

Four hyperparameters are tuned, the *learning rate*, the *number of trees*, maximum *tree depth*, and the *number of features* sampled at each split. The learning rate acts as a shrinkage parameter, limiting the weight the boosted model assigns to each tree in the ensemble (Hastie et al., 2006). The higher it is, the higher the chance of overfitting, but lower chance of a sub-optimised model (Hastie et al., 2006). Conversely, lower learning rates tend to render more generalisable models, but also bear the risk of a model with sub-optimal prediction power (Hastie et al., 2006). The number of trees specifies how many trees should be used in the ensemble (Hastie et al., 2006). Much like the learning rate, the more trees, the more learning the model does, but at the risk of overfitting (Hastie et al., 2006). The tree depth limits the number of splits each tree has (Hastie et al., 2006). The depth of a tree allows the algorithm to learn the interaction effects between features. The lower it is, the less complex relationships are learned (Hastie et al., 2006). The number of features instructs the algorithm how many of the features to select, at random, every time a tree is modelled (Hastie et al., 2006). Lower numbers minimise the chance that one feature masks the effect of another, when correlated (Hastie et al., 2006). Lastly, the splits were limited to a minimum of 10 observations.

The hyperparameter tuning process is performed in two stages. In the first stage, 20 models are 10-fold cross-validated on the training subset, using a randomly generated hyperparameter grid. Hence, each hyperparameter setup is trained 10 times. The average performance, as measured by the area under the curve (AUC), is computed for each of the 20 setups. Then, the results are visually inspected, via hyperparameter-performance plots (see Figure 23 in the next section for an example), and the best performing hyperparameter ranges are noted and used to inform the second stage.

Since the hyperparameter grid was set at random and only 20 combinations (of 4 hyperparameters) were generated, it is likely that one hyperparameter's performance may have been influenced by another hyperparameter. Thus, the hyperparameters' performance serves as a rough guideline for further fine-tuning the models.

In the second stage, the hyperparameters are also randomly generated, but they are now restricted to within the ranges identified in the first stage, thus focusing on fine-tuning the hyperparameters in order to identify the best performing hyperparameter setup. Sixteen models are 10-fold cross-validated on the training subset to this end.

Finally, the XGBoost algorithm, with best performing hyperparameter setup, is used to train the model on the entire training subset (downsampled). This final model is then used to produce predictions on the test subset, evaluate final performance, and generate the explanatory visualisations.

#### 3.2.2.2 Models fitted

In order to validate SRQ1, seven models are fitted using the above-mentioned hyperparameter tuning procedure. Three of the models are fitted on each of the clickstream data types - one containing the static features alone, one containing the temporal features alone, and one containing the sequential features alone. All features' types are identified in Appendix B, Figure 11. Three of the models are fitted on combinations of two types of clickstream features, one on static and temporal features, one on static and sequential features and another on temporal and sequential features. The seventh model is fitted on all 15 features (5 static, 4 temporal and 6 sequential).

### 3.3.3 Explanation visualisations

As mentioned in the Related Work section, two types of visualisations are used to illustrate differences in clickstream behaviour between converting and non-converting patterns – the feature importance plot and the feature effect plot. To compute the feature importance and effect, the so-far-unseen, test data subset is used.

The feature importance plot is generated with the vip package (Greenwell et al., 2020). The package's variable importance procedure follows the permutation methodology proposed by Fisher et al. (2019). Additionally, the package allows for the permutation procedure to be carried out multiple times, and the variable importance is returned as the average results of the permutation procedures. For this project the procedure was repeated ten times.

The feature effect plots are generated using the iml package (Molnar et al., 2018). A plot is produced for each of the features, with the number of intervals (over which the values are calculated) set to 20 – iml's default value. None of the plots looked 'shaky', so no optimisation on the number of intervals was performed.

## 4. Results

Two main aspects of the analysis are reported in this section. First, the results rendered by each of the seven fitted models are briefly discussed. Second, the explanatory visualisations are presented.

### 4.1 Model tuning and performance

The performance rendered by each of the seven fitted models is discussed in this subsection, whereby their respective performance is compared and contrasted.

### 4.1.1 Model performance by clickstream feature types

Seven models were fitted, three for each of the three clickstream feature types, and four made up of combinations of feature types, as described in subsection 3.2.2. Each models' hyperparameter setup was 10-fold cross validated on the training subset in both fitting stages. The target variable is a session's binary purchase outcome (buy or no buy).

The first stage of the hyperparameter tuning process involved 20 models with randomly generated hyperparameters. The first stage's performance was used to inform how the hyperparameters' ranges should be limited during the second stage of the fitting process. Figures 16 to 22, in Appendix D, provide detailed descriptions of each hyperparameter setup and its performance, for each of the seven fitted models. The figures also describe how the hyperparameter ranges were restricted for the second stage of the hyperparameter setup.

Aside from the tables illustrated in Figures 16 to 22 (Appendix D), four plots were produced (one for each hyperparameter tuned), as illustrated in Figure 23 below. This was in order to facilitate a visual inspection informing on how the models' performance changes as the hyperparameters change. During the first stage, the minimum and maximum performance across the seven models ranged in the AUC score from 0.5 (chance performance) to 0.7833. Figure 31 illustrates each of the models' first stage scores below.

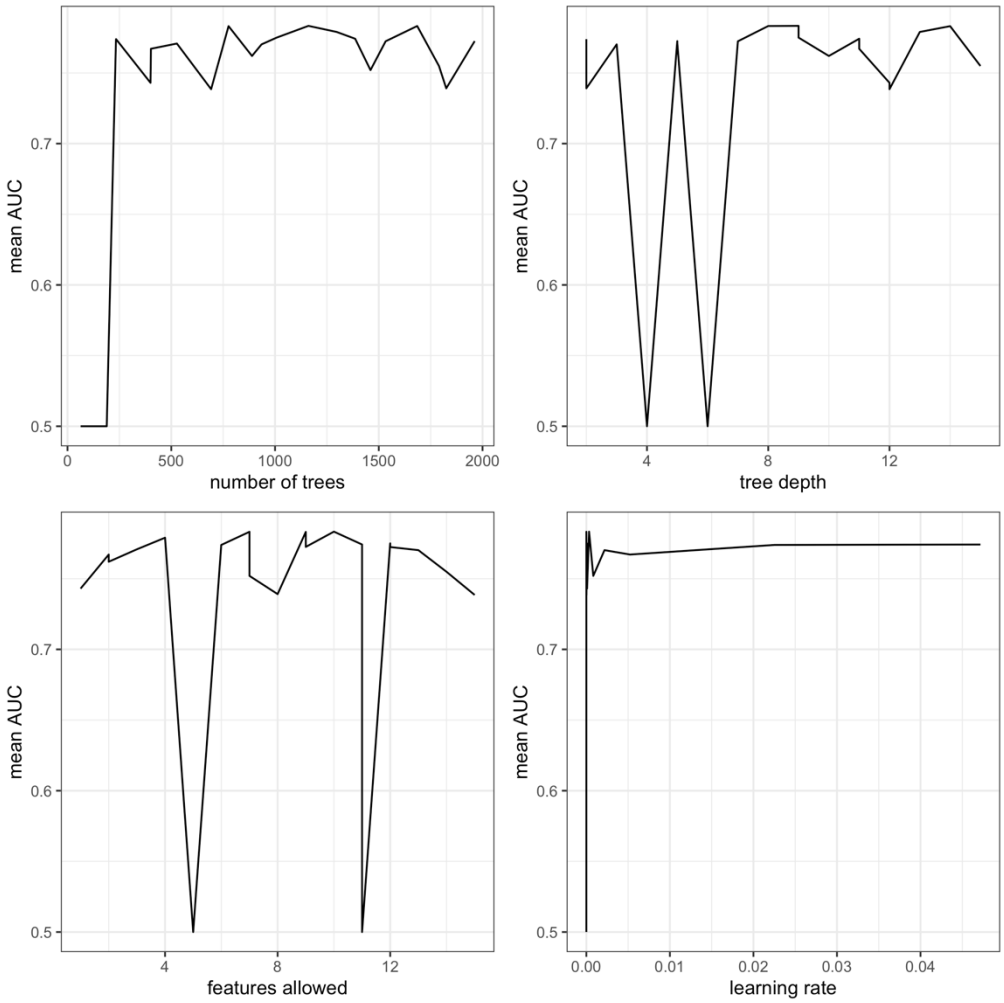


*Figure 23. First stage hyperparameter grid performance –* ***static, temporal and sequential features model (all features model)****. The graphs above illustrate how performance varied with the hyperparameter value. The hyperparameter values are randomly generated and performance is 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). Note that the hyperparameter combinations were generated at random, meaning that one hyperparameter's observed performance may have been influenced by another hyperparameter's value. These graphs serve as guidelines only, in order to limit the range allowed in the second stage of the hyperparameter tuning process. From the graphs, the hyperparameter were limited to the following ranges:*

- *Number of trees: between 700 and 1400*
- *Tree depth: between 5 and 10*
- *Features allowed: between 7 and 9*
- *Learning rate: between 0.005 and 0.03*

*NOTE: Six more similar figures were produced from the tables (illustrated in Figures 16 to 22), in order to inform the second stage of the hyperparameter tuning process, for each of the six remaining models.*

For the second hyperparameter tuning process, sixteen hyperparameter combinations were fitted for each of the seven models. The hyperparameter grid was randomly generated again, but this time limited to within the best performing ranges identified during the first hyperparameter tuning stage, described in subsection 3.2.2. Figures 24 to 30, in Appendix E, provide detailed descriptions of each hyperparameter setup and its respective performance during the second hyperparameter tuning stage, for each of the seven models. Note that the aforementioned figures also illustrate the best-performing hyperparameter setup for each model. In the second stage of the hyperparameter fitting stage, the fitted models' performance ranged in AUC score from 0.6544 to 0.7987 across the models, as illustrated in Figure 31 below.

| | **First hyperparameter tuning stage** | | **Second hyperparameter tuning stage** | | | |
|---|---|---|---|---|---|---|
| **Feature type** | **Min. ROC-AUC** | **Max. ROC-AUC** | **Min. ROC-AUC** | **Max. ROC-AUC** | **Number of features** | **Best model's test ROC-AUC** |
| **Static** | 0.5 | 0.7575 | 0.7561 | 0.7601 | 5 | 0.7646 |
| **Temporal** | 0.5 | 0.669 | 0.6423 | 0.6544 | 4 | 0.6674 |
| **Sequential** | 0.5 | 0.7314 | 0.7313 | 0.7354 | 6 | 0.7488 |
| **Static and temporal** | 0.5 | 0.7748 | 0.7753 | 0.7798 | 9 | 0.7853 |
| **Static and sequential** | 0.5 | 0.7782 | 0.7874 | 0.7907 | 11 | 0.8018 |
| **Temporal and sequential** | 0.5 | 0.7475 | 0.7462 | 0.7528 | 10 | 0.7658 |
| **All feature types** | 0.5 | 0.7833 | 0.7754 | 0.7987 | 15 | 0.8072 |

*Figure 31. Model performance by hyperparameter tuning stage. The table illustrates the minimum and maximum performance, as measured by the ROC-AUC, for each of the seven models trained, along with the feature type(s) and the corresponding number of features. The test performance of a model's best performing hyperparameter setup is also shown, as shown in the last column.*

Overall, all the tuned models were skilled to predict conversions well above chance level. The temporal clickstream features rendered the weakest performance among the seven models, while the model containing all feature types (that is static, temporal and sequential) rendered the best performance. Note that the differences between the tuned models' performance on the 10-fold cross-validated train subset is coherent with the differences in performance rendered when the tuned models were tested on the test subset. This indicates that the differences in performance identified are robust and generalisable, at least to this dataset.

Figure 32 below illustrates each of the fitted models' performance on the test data subset. The models are sorted by performance in ascending order from left to right. From the figure, the temporal data rendered the poorest performance at predicting conversion, followed by the sequential and the static features.

Looking at the models fitted on combinations of different clickstream feature types, the temporal and sequential combination of features rendered the poorest performance, followed by the static and temporal, and the static and sequential combinations. Lastly, the model fitted on all feature types (static, temporal and sequential) rendered the best performance. Note that the performance of the static and sequential combination of features is close to the performance rendered by the model fitted on all feature types. Also note that the model fitted on the static features renders comparable performance to the model fitted on the combination of temporal and sequential features.

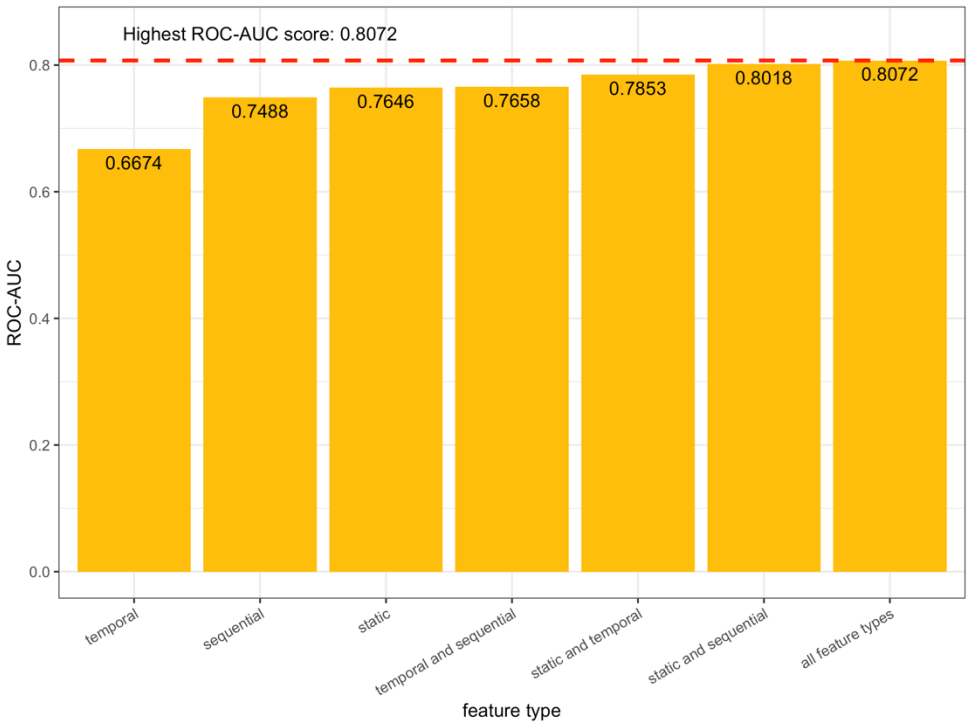


*Figure 32. Models' test performance at predicting a purchase (conversion). The barplot illustrates each of the fitted and tuned models' performance on the test subset of the clickstream data. The models are denoted by the feature types used and are sorted from least to best performing, as measured by the ROC-AUC. The least performing model was fitted on the temporal features of the clickstream dataset, while the best performing model was fitted on all the feature types (static, temporal and sequential).*

Lastly, Figures 33 and 34 below illustrate the seven models' ROC curve and precision-recall plots, respectively. Figure 33 suggests that none of the models rendered any differences in specificity-sensitivity trade-off, while the precision-recall plot in Figure 34, suggests that none of the models rendered any differences in precision-recall trade-off, and further confirms that all of the models are skilled to predict purchases from the clickstream data well above chance level.

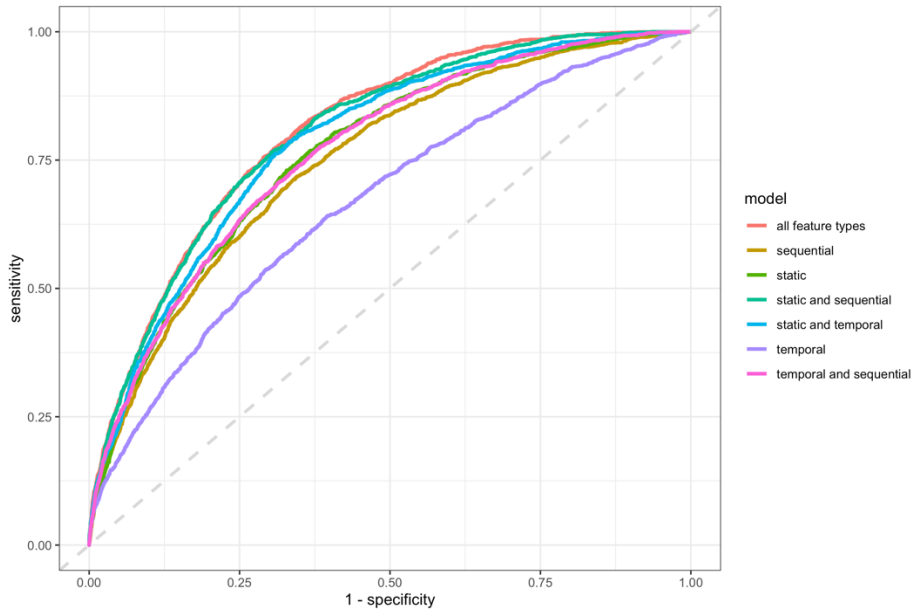


*Figure 33. The seven models' ROC curve plots. The plots illustrate each of the fitted and tuned models' performance on the test subset of the clickstream data. The models are denoted by the feature types used and are sorted from least to best performing, as measured by the ROC-AUC. The least performing model was fitted on the temporal features of the clickstream dataset, while the best performing model was fitted on all the feature types (static, temporal and sequential).*

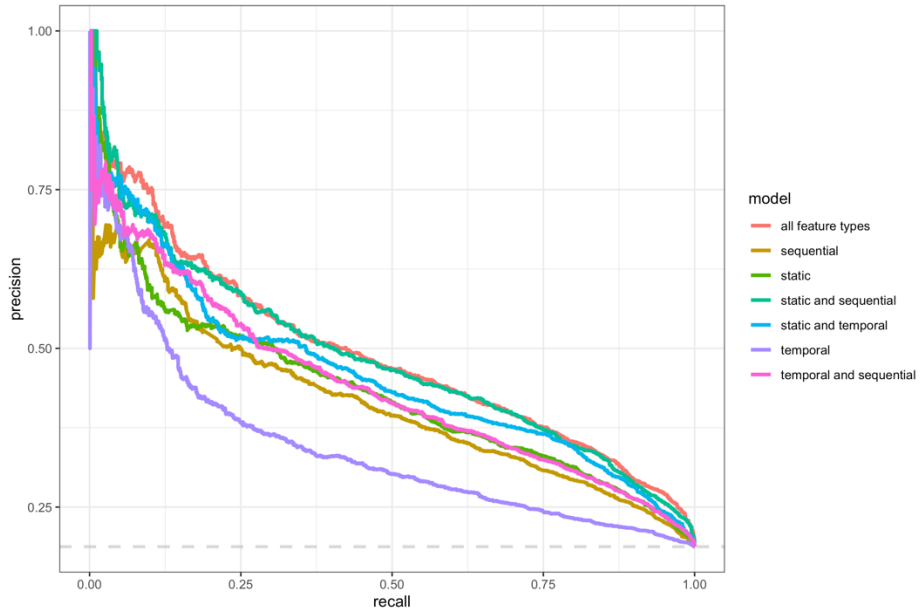


*Figure 34. The seven models' precision-recall plots. The precision-recall illustrate each of the fitted and tuned models' performance on the test subset of the clickstream data. A precision of 0.188, illustrated by the dashed line, indicates chance performance at predicting purchase events on the test data subset. All models make predictions which are above chance-performance level.*

### 4.2 Explanatory visualisations

The explanatory visualisations are extracted from the best-performing model – the model fitted on all feature types, 15 features in total. Five features were static, four temporal, and six sequential. Two types of explanatory visualisations are produced – a feature importance plot and 15 feature effect plots.

The feature importance plot, shown in Figure 35 below, illustrates the 15 features' importance in predicting the purchase outcome of a session. Features with higher importance values are, therefore, more important in predicting conversion. The feature importance plot allows UX professionals to attribute importance (what attribution) to each of the features used in the modelling process.

The figure illustrates that the five most important features are static and sequential in nature. Note that this is in line with the finding illustrated in Figure 32, which illustrates that the static and sequential features render the best predictive performance. The following five features seem noticeably less important for predicting the purchase outcome. The least important five features are a combination of a static, a temporal, and three sequential features.

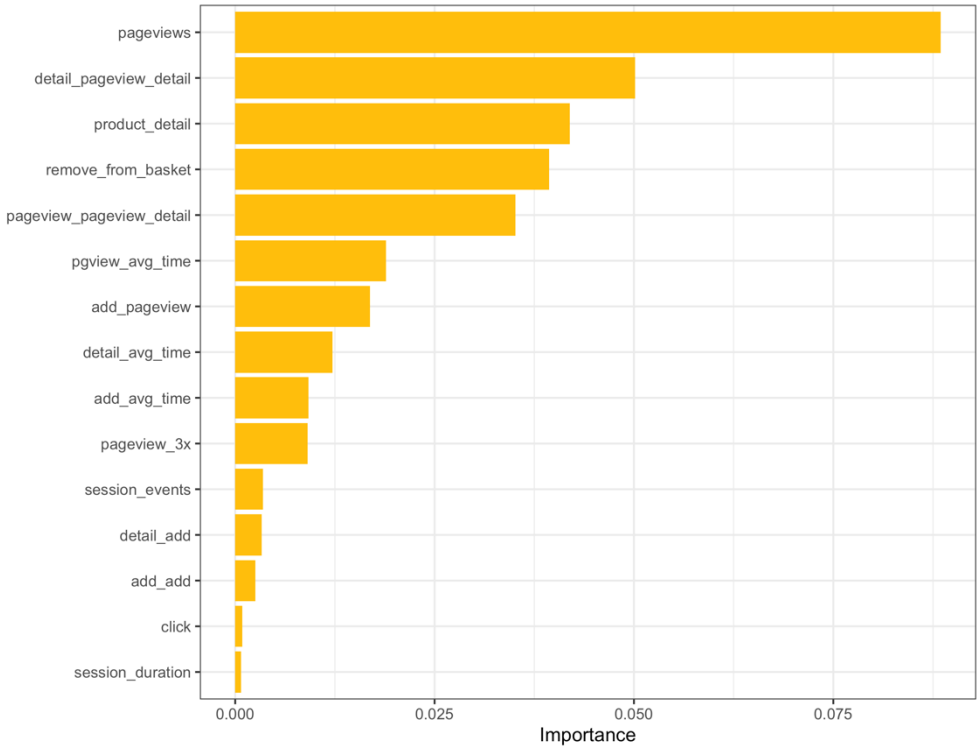


*Figure 35. Feature importance plot.*

The variable effect plots are produced, illustrated and discussed next. Figures 36, 37 and 38, below, present the static, temporal and sequential features' effect plots, respectively.

From Figure 36, the model predicts that the more pageviews (the most important predictor) take place in a session, the more likely it is that a purchase is made during the

session. The effect seems to plateau after about 50 pageviews. On the other hand, the more products' details are viewed (the third most important predictor), the less likely it is that a purchase is made during the session. Then, sessions where some items were removed from basket are more likely to lead to a purchase. This is especially interesting since the remove from basket event is the fourth most important predictor. Lastly, events per session and the number of indeterministic clicks seem to give somewhat mixed and uninformative results. The number indeterministic clicks seem to predict a purchase at first, but this effect drops just as sharply after two to three click events occur. The events per session feature seems to predict a purchase when few events occur, or many. Note that, since these two features are amongst the least important predictors, their interpretation may not provide insights even if their results appeared more informative.

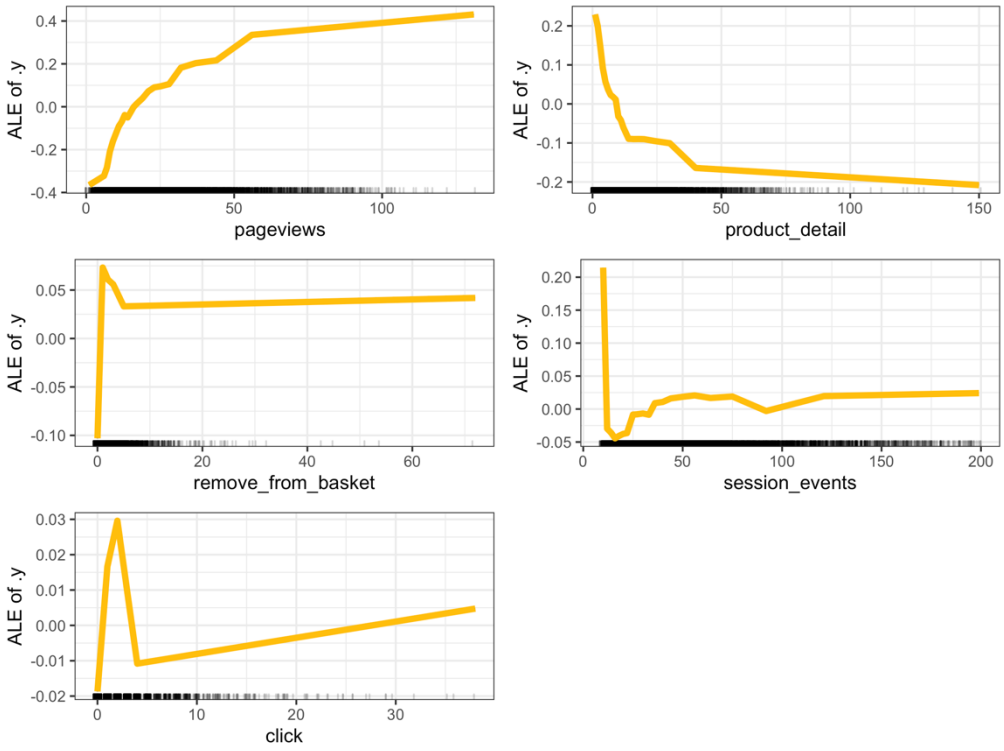


*Figure 36. Static features' ALE effect plots.*

From Figure 37, the model predicts that the higher the pageview dwell time is, the more likely a session is to end with a purchase. Note that although this effect drops slightly as the dwell time increases, the rug plot below the line illustrates that the number of observations rapidly diminishes after dwell time reaches about 200 seconds – thus, the results where few observations are available should be interpreted with caution, if at all. Next, the time spent on viewing a product's detail page rapidly and negatively impacts the likelihood that a purchase is made. This effect is especially interesting because the number of product detail views also impacts the likelihood of a purchase negatively. The add to basket dwell time and a session's duration seems to be rendering mixed results. This is perhaps negligible, given the two features' importance.

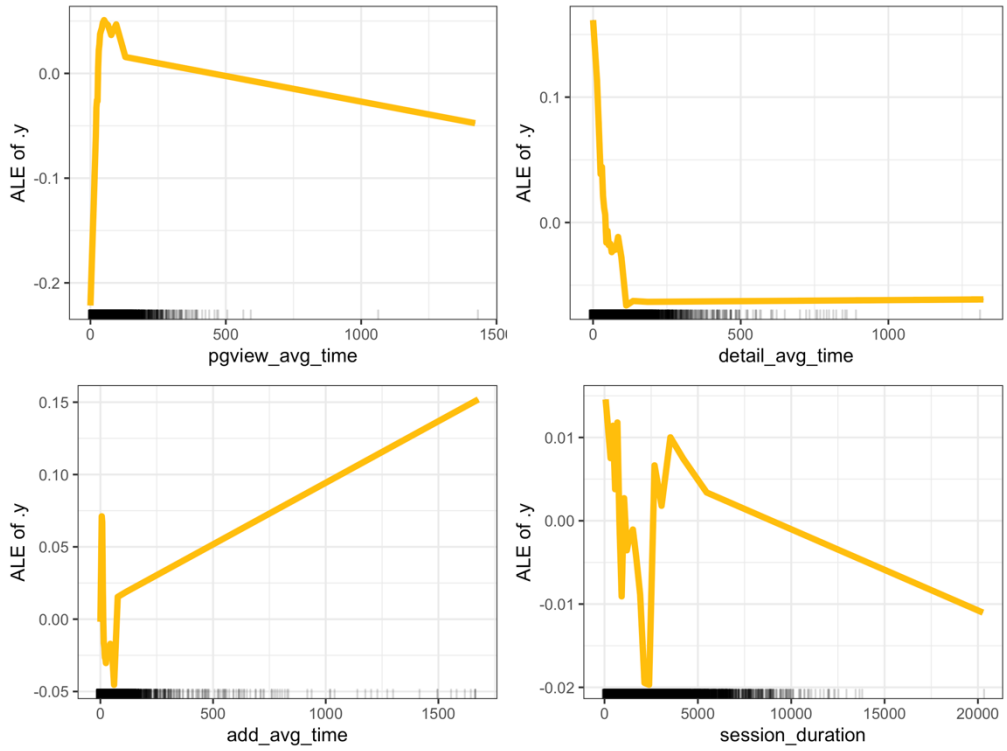


*Figure 37. Temporal features' ALE effect plots.*

From Figure 38, the *product detail* > *pageview* > *product detail* clickstream sequence, the *pageview* > *pageview* > *product detail* sequence, and the > *product detail* > *add to basket* sequence negatively impact the likelihood that a purchase is made during a session. These findings link with the previous findings that product detail pageviews and product detail dwell time impact the likelihood to purchase negatively. The *add to basket* > *pageview*, as well as the subsequent addition of items to basket positively impact the likelihood that a purchase is made during the session. Lastly, subsequent pageviews also impact the likelihood that a purchase is made positively, which is in line with the finding that multiple pageviews tend to lead to a purchase (from Figure 36).

Altogether, the feature effect plots tend to produce overlapping results between the feature types. That is, findings related to the same areas of the website can be represented in the static, temporal and sequential aspects of clickstream data. For example, Figure 36 illustrates that more pageviews (a static feature) are negatively associated with conversion, which is also reflected in Figure 38, which shows that the *product detail* > *pageview* > *product detail* sequence (a sequential feature) is negatively associated with conversion.

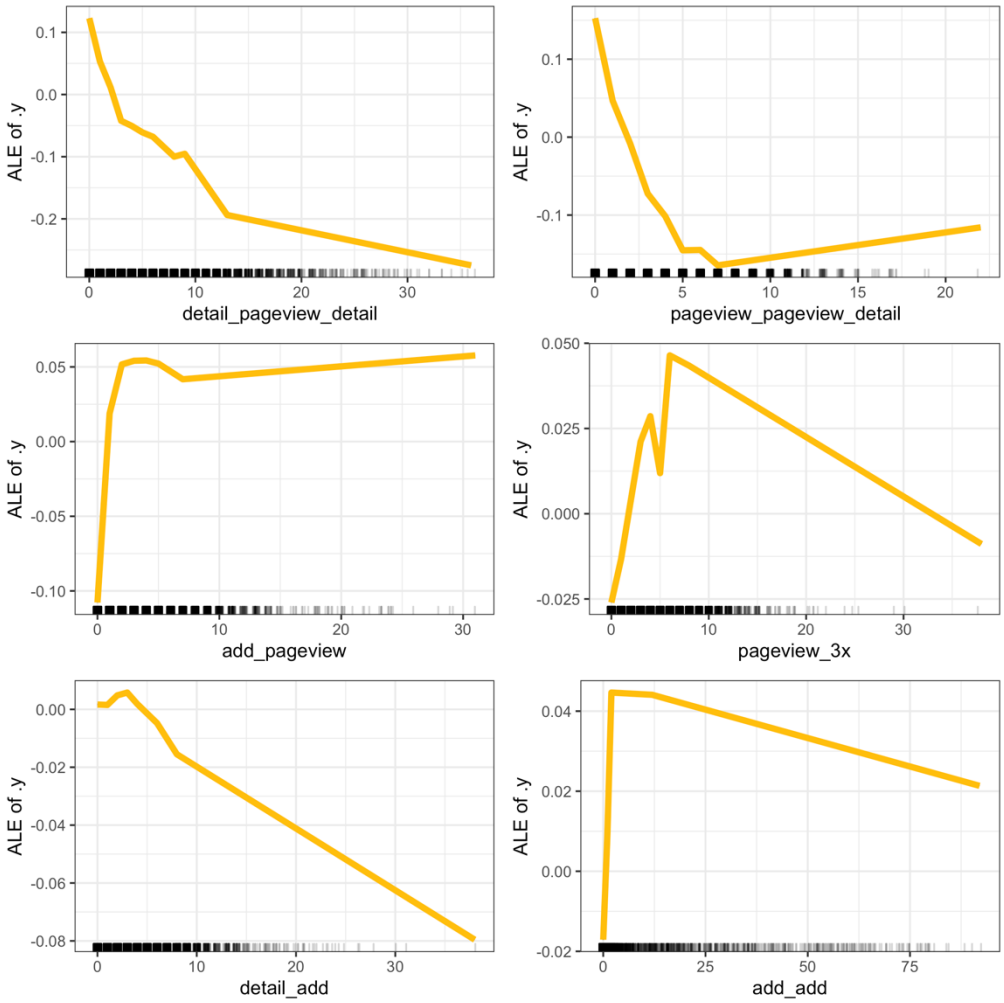


*Figure 38. Sequential features' ALE effect plots.*

## 5. Discussion

In this section the researched questions posed in this study are addressed first. Then, limitations and future studies are briefly discussed.

### 5.1 Addressing the research questions

This main research question addressed in this study is:

> *RQ: How can clickstream behavioural patterns, as identified by machine learning algorithms, be visualised in order to identify website (dis)satisfaction areas on an e-commerce website?*

This question, in turn, gave rise to three sub-questions. The nature of the study implies that the three research sub-questions have to be addressed before the main research question can be addressed. Each the SRQ's is addressed individually next.

> *SRQ1: Which aspects of clickstream data can predict conversion better? That is, do the static, temporal or sequential aspects of clickstream data help predict conversion? If so, which of the types renders the best performance? Does using specific combinations of static, temporal and sequential data outperform a model trained on a single clickstream data type? Does using all three data types render better performance?*

Seven models were fitted and contrasted in order to answer SRQ1. Three of the models were fitted on individual aspects of clickstream data - static, temporal and sequential aspects that is. Then, combinations of the three clickstream data aspects were fitted as four separate models – three considering sets of two feature types, such as static and temporal, and one considering all of them together.

The results highlighted that all feature types predict conversion above chance level, as illustrated in Figures 32 to 34. Then, as highlighted in Figure 32, the results indicate that, of all feature types, the static aspects of clickstream data can predict conversion best, followed by the sequential and temporal aspects respectively. All models fitted on combinations of static, temporal and sequential data rendered better performance than fitting a model on a single aspect of clickstream data. Additionally, making use of all aspects of clickstream data rendered the best predictive performance.

However, the differences in predictive performance between the different aspects of clickstream data were not always large. While making use of all aspects of clickstream data rendered the best predictive performance, the combination of static and sequential features rendered comparable performance (an AUC of 0.8072 for all data types, compared with an AUC of 0.8018 rendered by the combination of static and sequential features). The difference between the model fitted on the static data alone, and the model fitted on the temporal and sequential data combined was also minuscule (an AUC of 0.7658 for the model fitted on the temporal and sequential features com, compared with an AUC of 0.7646 rendered by the model fitted on the static features).

As an additional evaluation, in order to assess the models' performance at different threshold levels, Figures 33 and 34 compared the specificity-sensitivity trade-off and the precision-recall trade-offs between the seven models. A visual inspection of the plots revealed no noticeable differences amongst the models. SRQ2 is answered next.

> *SRQ2: What criteria should be used to assess how good an application aimed at illustrating the (non-)converting patters and, by extension, website satisfaction is? Specifically, what information should visualisations convey in order to illustrate what the differences between the converting and non-converting patterns are? And, under what circumstances does contrasting these patterns illustrate website (dis)satisfaction?*

To inform SRQ2, existing literature was consulted. Therefore, the assessment framework illustrated in Figure 4 describes how XAI visualisations should be illustrated in order to allow people, and UX professionals by extension, to identify the differences between converting and non-converting clickstream behavioural patterns. Specifically, the framework proposes that an XAI methodology is suitable to illustrate such (non-)converting patterns on an e-commerce website successfully, if:

- *The explanations are such that UX professionals can contrast converting and non-converting patterns,*

- *The explanations are such that UX professionals can identify the most important behavioural aspects (expressed as features) which lead to conversion, and*
- *The explanations are such that UX professionals can identify how a feature impacts the prediction outcome.*

Lastly, the link between conversion and website (dis)satisfaction was also informed by the literature. As illustrated in Figure 1 (and discussed in sub-section 2.1), assessing sessions where an intention to purchase is present provides an acceptable trade-off between losing information related to website satisfaction and removing the noise accompanied by visits where no purchase intent was present. This is because people visit e-commerce platforms for multiple reasons other than to make a purchase and, unless information related to these alternative reasons is removed, contrasting converting against non-converting patterns is more likely to reveal differences in behaviour related to, say, window shopping versus purchasing, rather than to illustrate website navigational patterns which are likely to alter website satisfaction levels. Thus, removing sessions without a purchase intent provides a context whereby the contrasted patterns are more likely be attributed to website satisfaction levels.

> *SRQ3: What XAI methodology, that adheres to the criteria identified in SRQ2, could be applied in order to allow UX professionals identify possible website (dis)satisfaction areas on an e-commerce website?*

The assessment framework identified is summarised in Figure 4, and is, to a large extent, reiterated above in order to answer SRQ2. Briefly, it states that explanations should allow UX professionals to contrast converting and non-converting patterns by illustrating which are the most important features in predicting conversion, and how each of the features used for modelling affect the likelihood of a conversion. It further states that sessions where no purchase intent was present should be removed in order to increase the chances that the patterns identified are related to website satisfaction levels, rather than to alternative visit reasons, such as price comparison, or window shopping. Lastly, the proposed framework indicates that static, temporal and sequential aspects of clickstream data should be considered.

The answer to SRQ3 was informed by existing literature and demonstrated through an application. In order to illustrate *which* are the most important features, this study proposes a model-agnostic methodology, as first described by Fisher et al. (2019).

Then, in order to illustrate *how* the features impact the prediction outcome, this study proposes the accumulated local effects (ALE) methodology for plotting feature effects, as originally described by (Apley & Zhu, 2016). This is because the ALE methodology is model agnostic, the results are robust to correlation effects, and the algorithm is computationally cheap.

In order to remove sessions where no purchase intent was present, this study proposes that sessions where no items were added to basket are removed. This is based on literature which identified that sessions where items were added to basket were considerably more likely to have a purchase intent.

Consideration towards the three clickstream data aspects were twofold. First, seven models were fitted on individual feature types and combinations thereof, and their respective predictive performance was compared and contrasted. Then, the model rendering the best predictive performance (the model fitted on all feature types, in this case) was used to produce the feature importance plot and the feature effects plots. Then, the plots were interpreted in order to compare and contrast the converting and non-converting patterns.

Answering the three sub-research questions culminates with an overarching answer, in the form of a proposed methodology, to the main research question: *How can clickstream behavioural patterns, as identified by machine learning algorithms, be visualised in order to identify website (dis)satisfaction areas on an e-commerce website?* This is summarised below, in short:

- *First, summarise the clickstream data at session level (sessionalise the data), such that static, temporal and sequential features are extracted,*
- *Then, identify, or use a proxy for purchase intent (such as the add to basket event) and remove all sessions without purchase intent,*
- *Next, fit multiple models on the sessionalised dataset in order to determine how the static, temporal and sequential aspects of clickstream data impact predictive performance, then*
- *Use a permutation-based algorithm to identify and plot the most important features, and*
- *Then, use the ALE methodology to illustrate how the features impact the model.*
- *Lastly, compare and contrast the identified patterns.*

This proposed methodology produced explanations which allow UX professionals to contrast patterns, identify important features (what attribution), and, identify how each of the features impact the prediction outcome (how attribution). And, since sessions without a purchase intent are removed from the data, the resulting patterns (and visualisations) are more likely to be attributed to website (dis)satisfaction.

From an academic perspective, both contributions filled a wide gap in the literature. Because of the limited literature on the topic, the assessment framework is the first of its kind, thus paving the way for future work to build upon it. Then, by linking the methodology to the assessment framework and current XAI visualisation methodologies, this paper contributed to the literature by providing a robust methodology for identifying complex clickstream behavioural patterns, as they relate to web satisfaction, and visualising them.

From a practitioner's perspective, this study's contributions are twofold. First, by developing a literature-backed assessment framework, future applications could be developed with these criteria as a guideline. This could inform practitioners wanting to assess such a tool, and developers aiming to develop one. Second, the proposed XAI methodology informs the development of future XAI applications aimed at illustrating website satisfaction levels. This methodology could be extended to websites other than e-commerce, or event to mobile application assessments.

### 5.2 Considerations, limitations and future work

#### 5.2.1 Assessment framework considerations and future studies

SRQ1 identified that some (combinations of) aspects of clickstream data are better at predicting conversion than others. The static clickstream features were the best individual clickstream data aspect at predicting performance, while the combination of static and sequential features produced predictive performance comparable to the model fitted on all feature types. Additionally, the ALE plots illustrated that there the insights derived from one aspect of clickstream data overlap with insights extracted from another. This suggests that not all aspects of feature data need to be used. Performance-wise, the static and

sequential aspects of clickstream data tend to render the best results. However, temporal clickstream data could reveal patterns which are relevant to UX professionals. In practice, performance and domain-specific considerations should be addressed, before developing such an application.

Next, the proposed assessment framework states that only sessions where there was a purchase intent should be analysed, so that differences between converting and non-converting behavioural patterns are illustrating web (dis)satisfaction areas. This in itself is a limitation both because website satisfaction could lead to purchase intent, and because, even if a session has purchase intent, visitors could decide not to make a purchase due to reasons other than website (dis)satisfaction. Such information is lost by applying the proposed methodology.

Lastly, in order to remove sessions without purchase intent, the add to basket event was used as a proxy, which is an imperfect measure of purchase intent. Future studies could experiment with different methodologies of identifying purchase intent, such as training a machine learning model to predict purchase intent.

### 5.2.2 Modelling limitations and future studies

Four main limitations are identified with the modelling.

First, the sequential information is limited to digrams and trigrams. Future studies could include more, or higher order, n-grams into the analysis. This could improve model performance and reveal interesting patterns.

Second, this study was limited to one class of machine learning algorithm. Existing literature finds that different classes of models identify different aspects of clickstream behaviour. Future studies could focus on comparing and contrasting the behavioural patterns identified by various classes of machine learning algorithms.

Third, due to time and computing resources constraints, limited model tuning was performed. Fine-grained hyperparameter tuning could have rendered better predictive performance. Experimenting with different sampling techniques, such as upsampling or leaving the dataset unbalanced, could have rendered better results also.

Lastly, interaction effects between variables were not considered for illustrative purposes in this study. Such illustrations would add a new dimension to the methodology and it should be considered in future studies.

### 5.2.3 Interpretation-related limitations and future studies

The explanations produced, have not been tested in practice. Future studies could test how useful these applications are for practicing UX professionals. Notably, in sub-section 4.2 the interpretations dealt with how the features impact the models but did not discuss whether such findings inform UX professionals about a website's (dis)satisfaction, or whether such insights are useful to UX professionals. While the existence of clickstream visualisation software and an identified need for better clickstream visualisation tools suggests that these insights might be useful, this should also be tested in practice.

## 6. Conclusion

This study began by arguing that, while the importance of a user's website satisfaction is becoming increasingly important, a wide gap in the literature regarding the visualisation of complex behavioural clickstream patterns, for website satisfaction levels, exists. It then proposed that, since machine learning is good at picking up underlying patterns in the data, such patterns could be visualised by applying XAI methodologies. However, since the

conjunction between XAI and UX design is yet an underexploited field of study, an assessment framework was informed and developed by reviewing existing literature on clickstream data applications and literature that links social sciences, human-computer interaction and XAI methodologies together.

The proposed framework, illustrated in Figure 4, proposes that XAI applications aimed at illustrating how clickstream behavioural patterns are linked to website satisfaction should consider the static (i.e. number of pageviews), temporal (i.e. product detail dwell time), and sequential (i.e. a user viewed the product detail and added it to basket, in that sequence) aspects of clickstream data. It then states that, in order to minimise the chance that the observed behavioural pattern contrast is due to reasons other than website satisfaction (such as price comparison), sessions without an intention to purchase should be removed. Lastly, it states that explanations should allow UX professionals to contrast the converting and non-converting patterns, identify which are the most important features, and how each of the features impacts the prediction outcome.

To illustrate how an application would fit within the framework, an application using clickstream data from a major European e-commerce fashion store was performed. To this end, seven XGBoost models were fitted on individual aspects of clickstream data and combinations thereof, in order to identify which aspects of clickstream data best predict conversion. The findings suggest that static clickstream data predicts conversion the best. Combining static data with other types of clickstream data renders considerably better predictions. In order to illustrate feature importance, a permutation-based algorithm was proposed. How the features impact the model was illustrated via ALE plots. This study concludes that the combination of feature importance plots and ALE plots allows UX professionals to contrast patterns, while illustrating what features impact conversion, and how they impact it. As long as sessions for which purchase intent was highly likely, linking conversion to website satisfaction levels is implicit.

The assessment framework paves the way for future studies to build upon it, while providing practitioners with an application-assessment tool. The proposed methodology provides practitioners with an example application to inform future developments and establishes best XAI practices for future studies to build upon.

## References

Apley, D. W., & Zhu, J. (2016). Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. *ArXiv Preprint ArXiv:11612.08468*. http://arxiv.org/abs/1612.08468

Bigon, L., Cassani, G., Greco, C., Lacasa, L., Pavoni, M., Polonioli, A., & Tagliabue, J. (2019). Prediction is very hard, especially about conversion*. Predicting user purchases from clickstream data in fashion e-commerce. *ArXiv Preprint ArXiv:1907.00400*. https://doi.org/10.1145/1234567890

Brainerd, J., & Blue, B. B. (2001). Case Study: E-Commerce Clickstream Visualization. *Information Visualization, IEEE Symposium*. http://courses.ischool.berkeley.edu/i247/s02/readings/brainerd.pdf

Cai, L., He, X., Dai, Y., & Zhu, K. (2018). Research on B2B2C E-commerce Website Design Based on User Experience. *Journal of Physics: Conference Series*, *1087*(6), 62043. https://doi.org/10.1088/1742-6596/1087/6/062043

Chandramohan, T. N., & Ravindran, B. (2018). A neural attention based approach for clickstream mining. *ACM International Conference Proceeding Series*, 118–127. https://doi.org/10.1145/3152494.3152505

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 785–794. https://doi.org/10.1145/2939672.2939785

Chen, T., He, T., Benesty, M., & Khotilovich, V. (2020). *Package "xgboost" Type Package Title Extreme Gradient Boosting*. https://doi.org/10.1145/2939672.2939785

Conglei Shi, Siwei Fu, Qing Chen, & Huamin Qu. (2015). VisMOOC: Visualizing video clickstream data from Massive Open Online Courses. *2015 IEEE Pacific Visualization Symposium (PacificVis)*, 159–166. https://doi.org/10.1109/PACIFICVIS.2015.7156373

Dias, J. P., & Ferreira, H. S. (2017). ScienceDirect The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017) Automating the Extraction of Static Content and Dynamic Behaviour from e-Commerce Websites. *Procedia Computer Science*, *109*, 297–304. https://doi.org/10.1016/j.procs.2017.05.355

Dix, A. (2017). Human–computer interaction, foundations and new paradigms. *Journal of Visual Languages and Computing*, *42*, 122–134. https://doi.org/10.1016/j.jvlc.2016.04.001

Dove, G., Halskov, K., Forlizzi, J., & Zimmerman, J. (2017). UX design innovation: Challenges for working with machine learning as a design material. *Conference on Human Factors in Computing Systems - Proceedings*, *2017-May*, 278–288. https://doi.org/10.1145/3025453.3025739

Econsultancy, & RedEye. (2018). *2018 Optimization Report - Econsultancy*. https://econsultancy.com/reports/2018-optimization-report/

Filipowska, A., Kaluzny, P., & Skrzypek, M. (2019). Improving User Experience in e-Commerce by Application of Process Mining Techniques. *Zeszyty Naukowe Politechniki Częstochowskiej Zarządzanie*, *33*(1), 30–40. https://doi.org/10.17512/znpcz.2019.1.03

Fisher, A., Rudin, C., & Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, *20*, 1–81. http://jmlr.org/papers/v20/18-760.html.

Frhan, A. J. (2017). Website Clickstream Data Visualization Using Improved Markov Chain Modelling in Apache Flume. *MATEC Web of Conferences*, *125*. https://doi.org/10.1051/matecconf/201712504025

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. In *Source: The Annals of Statistics* (Vol. 29, Issue 5).

Gleason, D. (2019). *The 2019 State of Conversion Optimization Report*. Conversionxl.Com. https://cxl.com/blog/2019-conversion-optimization-report/

Greenwell, B., Boehmke, B., & Gray, B. (2020). *Package "vip."* https://doi.org/10.1007/s10115-013-0679-x

Gudigantala, N., Bicen, P., & Eom, M. (Tae in). (2016). An examination of antecedents of conversion rates of e-commerce retailers. *Management Research Review*, *39*(1), 82–114. https://doi.org/10.1108/MRR-05-2014-0112

Hartono, E., & Holsapple, C. W. (2019). Website visual design qualities: A threefold framework. *ACM Transactions on Management Information Systems*, *10*(1). https://doi.org/10.1145/3309708

Hastie, T., Friedman, J., & Tibshirani, R. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Predictions* (2nd ed.). Stanford. https://web.stanford.edu/~hastie/ElemStatLearn/

Hastie, T., Tibshirani, R., James, G., & Witten, D. (2006). An Introduction to Statistical Learning, Springer Texts. In *Springer Texts* (Vol. 102). https://doi.org/10.1016/j.peva.2007.06.006

Kateja, R., Rohith, A., Kumar, P., & Sinha, R. (2014). VizClick visualizing clickstream data. *2014 International Conference on Information Visualization Theory and Applications (IVAPP)*, 247–255.

Koehn, D., Lessmann, S., & Schaal, M. (2020). Predicting online shopping behaviour from clickstream data using deep learning. *Expert Systems with Applications*, *150*, 113342. https://doi.org/10.1016/j.eswa.2020.113342

Kohavi, R., Mason, L., Parekh, R., & Zheng, Z. (2004). Lessons and challenges from mining retail e-commerce data. In *Machine Learning* (Vol. 57, Issues 1-2 SPEC. ISS., pp. 83–113). https://doi.org/10.1023/B:MACH.0000035473.11134.83

Krause, J., Perer, A., & Ng, K. (2016). Interacting with predictions: Visual inspection of black-box machine learning models. *Conference on Human Factors in Computing Systems - Proceedings*, 5686–5697. https://doi.org/10.1145/2858036.2858529

Liu, Z., Dev, H., Dontcheva, M., & Hoffman, M. (2016). Mining , Pruning and Visualizing Frequent Patterns for Temporal Event Sequence Analysis. *Workshop on Temporal & Sequential Event Analysis*, 2–4. https://doi.org/10.1109/ICDE

Liu, Z., Wang, Y., Dontcheva, M., Hoffman, M., Walker, S., & Wilson, A. (2017). Patterns and Sequences: Interactive Exploration of Clickstreams to Understand Common Visitor Paths. *IEEE Transactions on Visualization and Computer Graphics*, *23*(1), 321–330. https://doi.org/10.1109/TVCG.2016.2598797

Lo, C., Frankowski, D., & Leskovec, J. (2016). Understanding behaviors that lead to purchasing: A case study of pinterest. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 531–540. https://doi.org/10.1145/2939672.2939729

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., & Lee, S.-I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, *2*(1), 56–67. https://doi.org/10.1038/s42256-019-0138-9

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *2017-Decem*, 4766–4775. https://github.com/slundberg/shap

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. In *Artificial Intelligence* (Vol. 267, pp. 1–38). https://doi.org/10.1016/j.artint.2018.07.007

Mokryn, O., Bogina, V., & Kuflik, T. (2019). Will this session end with a purchase? Inferring current purchase intent of anonymous visitors. *Electronic Commerce Research and Applications*, *34*. https://doi.org/10.1016/j.elerap.2019.100836

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

Molnar, C., Casalicchio, G., & Bischl, B. (2018). iml: An R package for Interpretable Machine Learning Software • Review • Repository • Archive. *The Journal of Open Source Software*. https://doi.org/10.21105/joss.00786

Narang, B., Trivedi, P., & Dubey, M. K. (2017). Towards an Understanding of UX (User Experience) and UXD (User Experience Design), an Applicability Based Framework for Ecommerce, Intranets, Mobile and Tablet and Web Usability. *International Journal of Advanced Research in Computer Science*, *8*(5), 2764–2768. www.ijarcs.info

Padidem, D. K., & Nalini, C. (2017). Process mining approach to discover shopping behavior process model in ecommerce web sites using click stream data. *International Journal of Civil Engineering and Technology*, *8*(1), 948–955. http://www.iaeme.com/IJCIET/index.asp948http://www.iaeme.com/IJCIET/issues.asp?JType=IJCIET&VType=8&IType=1http://www.iaeme.com/IJCIET/issues.asp?JType=IJCIET&VType=8&IType=1

R-project.org. (2017). *Fast n-Gram "Tokenization."* https://github.com/wrathematics/ngram

Raphaeli, O., Goldstein, A., & Fink, L. (2017). Analyzing online consumer behavior in mobile and PC devices: A novel web usage mining approach. *Electronic Commerce Research and Applications*. https://doi.org/10.1016/j.elerap.2017.09.003

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Model-Agnostic Interpretability of Machine Learning. *ArXiv:1606.05386v1*. http://arxiv.org/abs/1606.05386

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215. https://doi.org/10.1038/s42256-019-0048-x

Saket, B., Moritz, D., Lin, H., Dibia, V., Demiralp, C., & Heer, J. (2018). *Beyond Heuristics: Learning Visualization Design*. https://kopoljs.github.io/

Sheil, H., & Rana, O. (2018). Classifying and Recommending Using Gradient Boosted Machines and Vector Space Models. *Advances in Intelligent Systems and Computing*, *650*, 214–221. https://doi.org/10.1007/978-3-319-66939-7_18

Su, Q., & Chen, L. (2014). *A method for discovering clusters of e-commerce interest patterns using click-stream data*. https://doi.org/10.1016/j.elerap.2014.10.002

*Tidymodels*. (n.d.). Retrieved May 7, 2020, from https://www.tidymodels.org/

*Tidyverse*. (n.d.). Retrieved March 6, 2020, from https://www.tidyverse.org/packages/

Wang, D., Yang, Q., Abdul, A., & Lim, B. Y. (2019). Designing theory-driven user-centric explainable AI. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3290605.3300831

Wang, G., Zhang, X., Tang, S., Zheng, H., & Zhao, B. Y. (2016). Unsupervised clickstream clustering for user behavior analysis. *Conference on Human Factors in Computing Systems - Proceedings*, 225–236. https://doi.org/10.1145/2858036.2858107

Wei, J., Shen, Z., Sundaresan, N., & Ma, K. (2012). Visual cluster exploration of web clickstream data. *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 3–12. https://doi.org/10.1109/VAST.2012.6400494

Zhao, J., Liu, Z., Dontcheva, M., Hertzmann, A., & Wilson, A. (2015). Matrixwave: Visual comparison of event sequence data. *Conference on Human Factors in Computing Systems - Proceedings*, *2015-April*, 259–268. https://doi.org/10.1145/2702123.2702419

Zhao, X., Louca, R., & Hu, D. (2020). *The Difference Between a Click and a Cart-Add : Learning Interaction-Specific Embeddings*. 454–460. https://doi.org/10.1145/3366424.3386197

**Appendix A: Wang et al. (2019) HCI-XAI Framework**

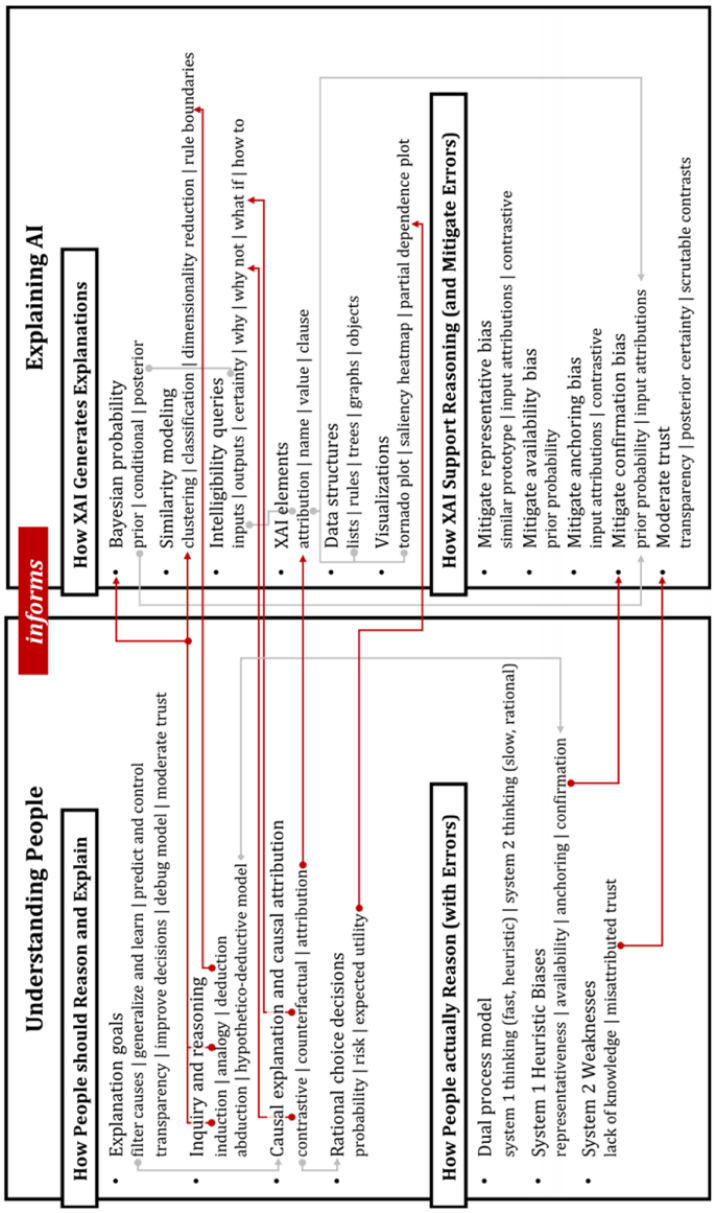


*Figure 3. Figure illustrating the links between various HCI theoretical aspects and XAI methodologies. Red arrows link appeals to specific aspects of human reasoning to existing XAI methodologies. Grey arrows illustrate the relationships between various reasoning processes and associations between XAI methodologies. Reprinted from* Wang et al. (2019, p. 4).

**Appendix B: Descriptive statistics**

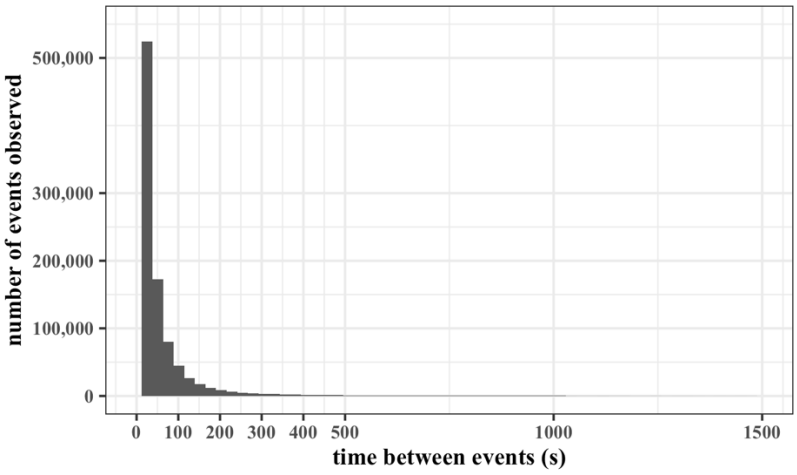


*Figure 9. Histogram illustrating the time between clickstream events*

| *n* | *n-gram event sequence* | *Freq. count* | *% of total* |
|---|---|---|---|
| *2* | *pageview > product detail* | 358,213 | 24.66% |
| *2* | *pageview > pageview* | 310,246 | 21.35% |
| *2* | *Product detail > pageview* | 280,618 | 19.32% |
| *2* | *Add to basket > add to basket* | 145,227 | 9.99% |
| *2* | *Product detail > add to basket* | 83,846 | 5.77% |
| *2* | *add to basket > pageview* | 78,584 | 5.41% |
| *3* | *pageview > product detail > pageview* | 250,069 | 17.62% |
| *3* | *product detail > pageview > product detail* | 202,821 | 14.23% |
| *3* | *pageview > pageview > pageview* | 190,531 | 13.43% |
| *3* | *add to basket > add to basket > add to basket* | 135,692 | 9.56% |
| *3* | *pageview > pageview > product detail* | 92,547 | 6.52% |

*Figure 10. Table illustrating the most frequent 2 and 3 event n-grams.*

| *Feature* | *Type* | *Mean* | *Std. Dev.* | *Min* | *Median* | *Max* |
|---|---|---|---|---|---|---|
| *Pageview* | Static | 22.23 | 16.46 | 1 | 17 | 134 |
| *Product detail* | Static | 13.89 | 12.98 | 1 | 10 | 173 |
| *Add to basket** | Static | 6.82 | 17.53 | 1 | 2 | 185 |
| *Remove from basket* | Static | 1.06 | 2.59 | 0 | 0 | 72 |
| *click* | Static | 0.61 | 2.25 | 0 | 0 | 40 |
| *Session duration (s)* | Temporal | 2035.52 | 1753.45 | 44 | 1514 | 20269 |
| *Session events* | Static | 44.61 | 34.69 | 10 | 33 | 199 |
| *Pageview avg time (s)* | Temporal | 49.37 | 44.24 | 0 | 36.06 | 1421 |
| *Detail avg time (s)* | Temporal | 69.82 | 65.69 | 0 | 53 | 1742 |
| *Add avg time (s)* | Temporal | 41.24 | 76.1 | 0 | 25 | 1755 |
| *Pageview> detail** | Sequential | 10.77 | 10.45 | 0 | 7 | 87 |
| *Pageview> pageview** | Sequential | 5.54 | 4.8 | 0 | 4 | 57 |
| *Detail> pageview** | Sequential | 8.34 | 9.19 | 0 | 5 | 83 |
| *Add>add* | Sequential | 2.09 | 8.73 | 0 | 0 | 92 |
| *Detail>add* | Sequential | 2.51 | 2.7 | 0 | 2 | 38 |
| *Add>pageview* | Sequential | 2.32 | 2.59 | 0 | 1 | 37 |
| *Pageview>detail>pageview** | Sequential | 4.77 | 5.24 | 0 | 3 | 46 |
| *Pageview 3x* | Sequential | 2.72 | 2.79 | 0 | 2 | 38 |
| *Detail>pageview>detail* | Sequential | 3.87 | 4.53 | 0 | 2 | 44 |
| *Add 3x** | Sequential | 1.29 | 5.77 | 0 | 0 | 61 |
| *Pageview>pageview>detail* | Sequential | 2.77 | 2.35 | 0 | 2 | 22 |

** = feature dropped before modelling due to high correlation (>0.9) with other features*.

*Figure 11. Summary statistics of the pre-processed dataset*

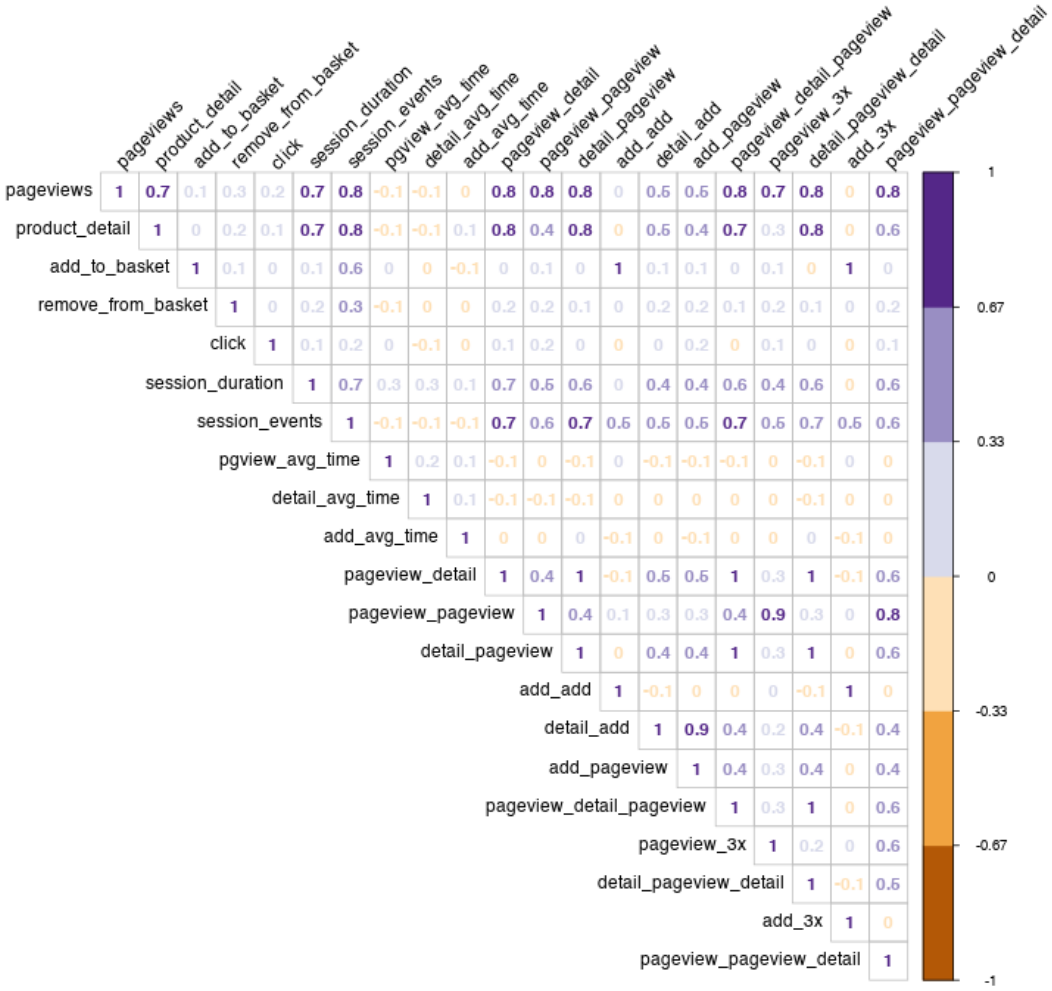


*Figure 12. Pearson correlations of the sessionalised dataset. The correlation matrix in this figure indicates the correlations between the explanatory variables, as observed on the training subset of the sessionalised dataframe. Some of the correlations are near-perfect (> 0.9), illustrating collinearity between them.*

**Appendix C: Contrasting probability distribution functions by purchase outcome**

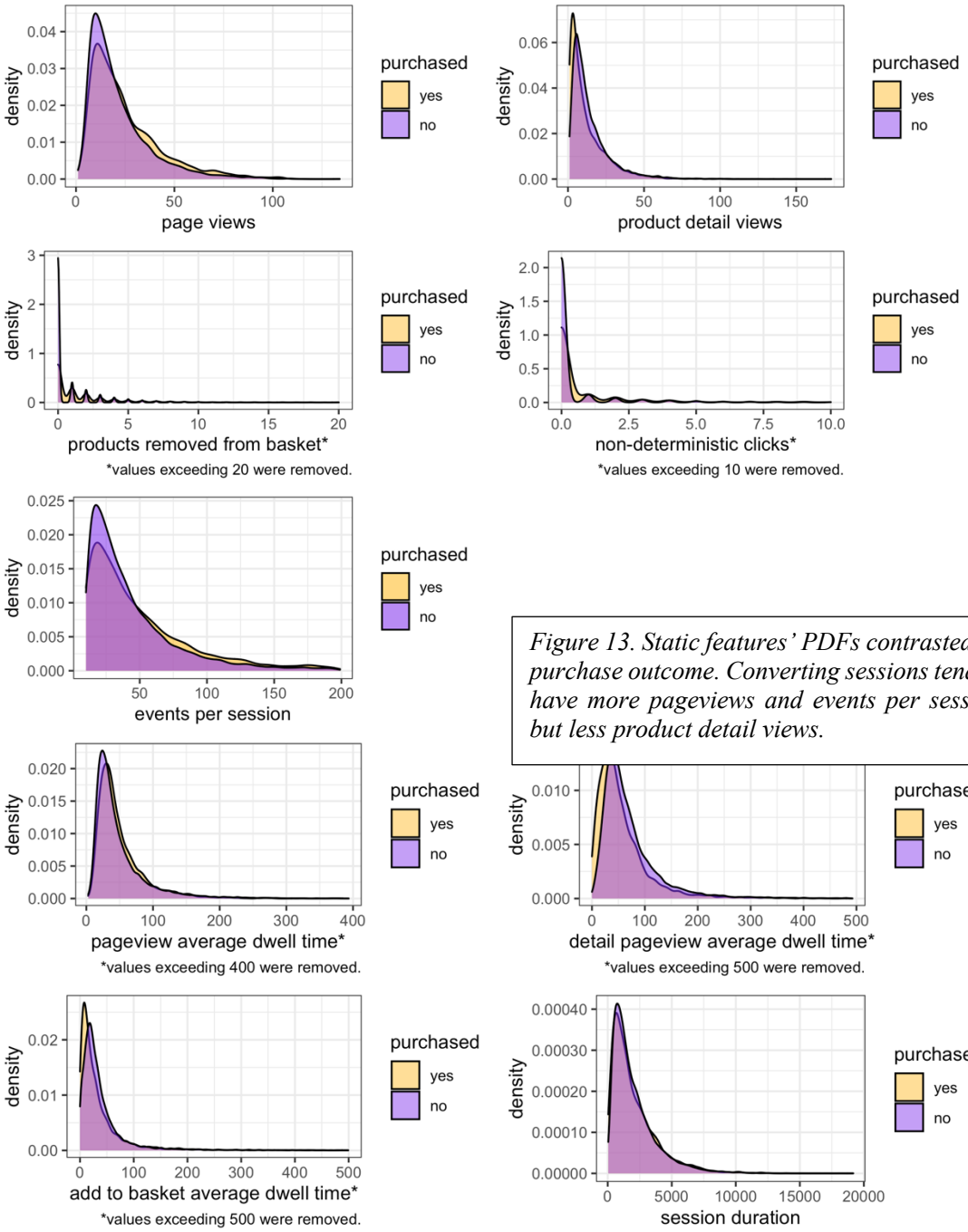


*Figure 13. Static features' PDFs contrasted by purchase outcome. Converting sessions tend to have more pageviews and events per session but less product detail views.*

*Figure 14. Temporal features' PDFs contrasted by purchase outcome. Converting sessions tend to spend less time viewing product details and the dwell time to add to basket is shorter, while pageviews tend to take longer.*

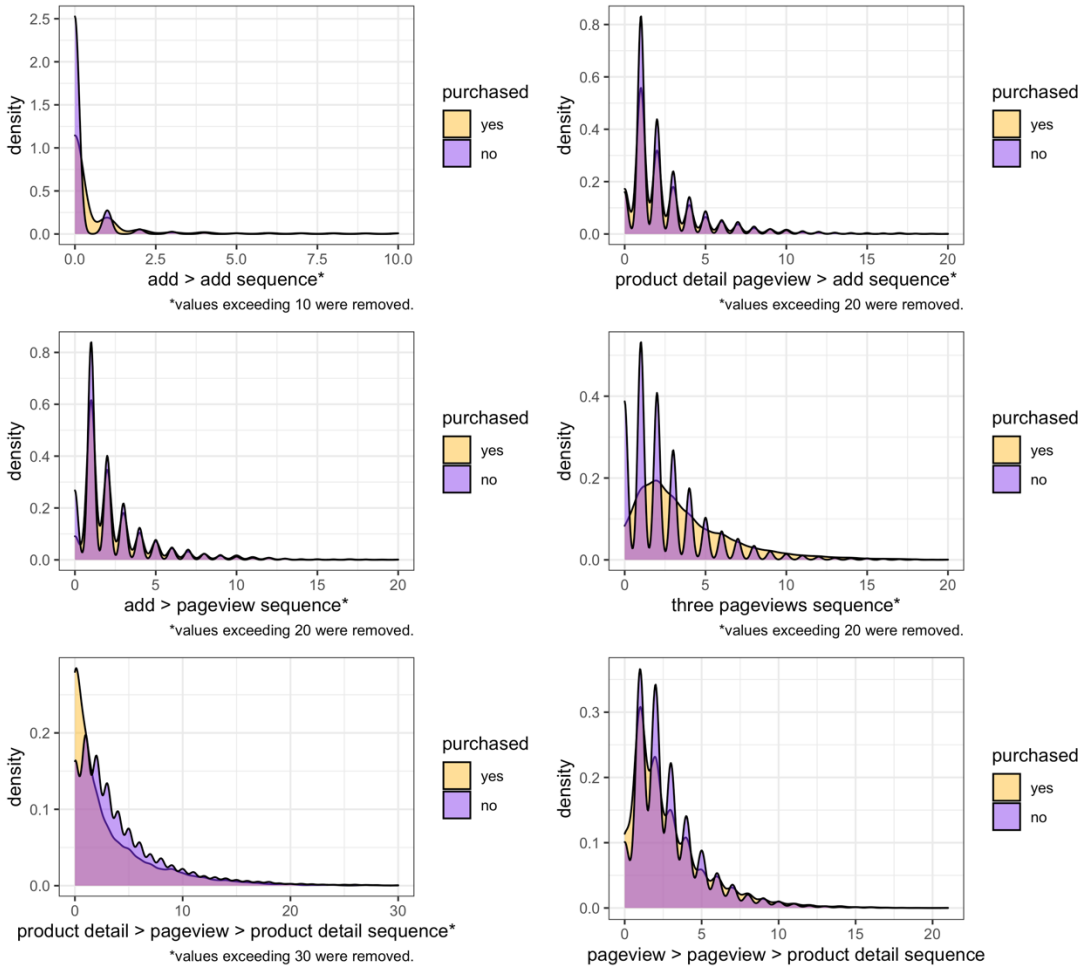


*Figure 15. Sequential features' PDFs contrasted by purchase outcome.*

**Appendix D: Hyperparameter tuning first stage results**

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 4 | 1161 | 9 | 0.0003 | 0.7575 | 10 | 0.0047 |
| 4 | 776 | 8 | 0.0001 | 0.757 | 10 | 0.0045 |
| 1 | 402 | 14 | 0.0052 | 0.7548 | 10 | 0.0049 |
| 3 | 1685 | 13 | 0.0000 | 0.7542 | 10 | 0.0044 |
| 5 | 1008 | 9 | 0.0001 | 0.7532 | 10 | 0.0046 |
| 5 | 1533 | 11 | 0.0001 | 0.7502 | 10 | 0.0048 |
| 4 | 1962 | 2 | 0.0001 | 0.7463 | 10 | 0.0047 |
| 3 | 234 | 5 | 0.0225 | 0.7456 | 10 | 0.0046 |
| 5 | 935 | 7 | 0.0022 | 0.745 | 10 | 0.0048 |
| 2 | 1297 | 5 | 0.0001 | 0.7425 | 10 | 0.0043 |
| 2 | 889 | 3 | 0.0001 | 0.7421 | 10 | 0.0045 |
| 1 | 400 | 11 | 0.0001 | 0.7394 | 10 | 0.0054 |
| 5 | 1386 | 10 | 0.0472 | 0.7392 | 10 | 0.0038 |
| 6 | 692 | 15 | 0.0001 | 0.7388 | 10 | 0.0056 |
| 6 | 1790 | 2 | 0.0001 | 0.7376 | 10 | 0.0057 |
| 2 | 527 | 12 | 0.0001 | 0.7345 | 10 | 0.0043 |
| 3 | 1460 | 2 | 0.0008 | 0.7241 | 10 | 0.005 |
| 3 | 1825 | 12 | 0.0001 | 0.7148 | 10 | 0.0049 |
| 2 | 189 | 4 | 0.0001 | 0.5 | 10 | 0 |
| 4 | 63 | 6 | 0.0001 | 0.5 | 10 | 0 |

*Figure 16. First stage hyperparamter grid results – **static features model**. This table illustrates the performance results (rounded to 4 decimal places) for the first stage of the hyperparameter tuning procedure, ordered by performance. Each of the 20 setups is randomly generated and 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). Based on these results, the hyperparameter grid was restricted during the second hyperparameter tuning stage as follows:*

- *Number of trees: between 750 and 1400*
- *Tree depth: between 7 and 9*
- *Features allowed: between 4 and 6*
- *Learning rate: >= 0.005*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 2 | 234 | 2 | 0.0225 | 0.669 | 10 | 0.0049 |
| 4 | 935 | 3 | 0.0022 | 0.6649 | 10 | 0.0046 |
| 3 | 1962 | 5 | 0.0001 | 0.663 | 10 | 0.0041 |
| 3 | 1533 | 7 | 0.0001 | 0.6602 | 10 | 0.0044 |
| 3 | 1161 | 9 | 0.0003 | 0.6592 | 10 | 0.0045 |
| 2 | 1460 | 2 | 0.0008 | 0.6581 | 10 | 0.005 |
| 1 | 402 | 11 | 0.0052 | 0.6567 | 10 | 0.0045 |
| 3 | 1008 | 9 | 0.0001 | 0.6566 | 10 | 0.0047 |
| 2 | 1685 | 8 | 0.0001 | 0.6515 | 10 | 0.0061 |
| 3 | 776 | 14 | 0.0001 | 0.6496 | 10 | 0.0044 |
| 1 | 527 | 5 | 0.0001 | 0.6476 | 10 | 0.0033 |
| 2 | 1825 | 2 | 0.0001 | 0.646 | 10 | 0.0053 |
| 2 | 1297 | 13 | 0.0001 | 0.6419 | 10 | 0.006 |
| 1 | 889 | 10 | 0.0001 | 0.6353 | 10 | 0.0036 |
| 1 | 400 | 12 | 0.0001 | 0.6342 | 10 | 0.0038 |
| 4 | 692 | 12 | 0.0001 | 0.6319 | 10 | 0.0038 |
| 4 | 1790 | 15 | 0.0001 | 0.6316 | 10 | 0.003 |
| 3 | 1386 | 11 | 0.0472 | 0.6269 | 10 | 0.0061 |
| 2 | 189 | 4 | 0.0001 | 0.5 | 10 | 0 |
| 3 | 63 | 6 | 0.0001 | 0.5 | 10 | 0 |

*Figure 17. First stage hyperparameter grid results – **temporal features model**. This table illustrates the performance results (rounded to 4 decimal places) for the first stage of the hyperparameter tuning procedure, ordered by performance. Each of the 20 setups is randomly generated and 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). Based on these results, the hyperparameter grid was restricted during the second hyperparameter tuning stage as follows:*

- *Number of trees: between 800 and 1200*
- *Tree depth: between 7 and 9*
- *Features allowed: between 1 and 2*
- *Learning rate: between 0.005 and 0.03*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 4 | 1161 | 9 | 0.0003 | 0.7314 | 10 | 0.004 |
| 4 | 776 | 14 | 0.0001 | 0.7309 | 10 | 0.0036 |
| 3 | 1685 | 8 | 0.0001 | 0.7305 | 10 | 0.004 |
| 3 | 234 | 2 | 0.0225 | 0.7296 | 10 | 0.0043 |
| 1 | 402 | 11 | 0.0052 | 0.7291 | 10 | 0.0038 |
| 4 | 1962 | 5 | 0.0001 | 0.7289 | 10 | 0.0044 |
| 5 | 1533 | 7 | 0.0001 | 0.7284 | 10 | 0.004 |
| 5 | 1008 | 9 | 0.0001 | 0.7276 | 10 | 0.0041 |
| 5 | 935 | 3 | 0.0022 | 0.7274 | 10 | 0.0044 |
| 2 | 1297 | 13 | 0.0001 | 0.7239 | 10 | 0.0039 |
| 2 | 889 | 10 | 0.0001 | 0.7237 | 10 | 0.004 |
| 2 | 527 | 5 | 0.0001 | 0.7215 | 10 | 0.0046 |
| 3 | 1460 | 2 | 0.0008 | 0.7189 | 10 | 0.0044 |
| 5 | 1386 | 11 | 0.0472 | 0.7131 | 10 | 0.0038 |
| 1 | 400 | 12 | 0.0001 | 0.7125 | 10 | 0.0039 |
| 6 | 692 | 12 | 0.0001 | 0.7123 | 10 | 0.0036 |
| 6 | 1790 | 15 | 0.0001 | 0.7114 | 10 | 0.0038 |
| 3 | 1825 | 2 | 0.0001 | 0.7044 | 10 | 0.0043 |
| 2 | 189 | 4 | 0.0001 | 0.5 | 10 | 0 |
| 4 | 63 | 6 | 0.0001 | 0.5 | 10 | 0 |

*Figure 18. First stage hyperparameter grid results – **sequential features model**. This table illustrates the performance results (rounded to 4 decimal places) for the first stage of the hyperparameter tuning procedure, ordered by performance. Each of the 20 setups is randomly generated and 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). Based on these results, the hyperparameter grid was restricted during the second hyperparameter tuning stage as follows:*

- *Number of trees: between 750 and 1400*
- *Tree depth: between 7 and 9*
- *Features allowed: between 5 and 6*
- *Learning rate: >= 0.005*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 6 | 776 | 14 | 0.0001 | 0.7748 | 10 | 0.0043 |
| 7 | 1161 | 9 | 0.0003 | 0.7734 | 10 | 0.0044 |
| 5 | 1685 | 8 | 0.0001 | 0.7733 | 10 | 0.004 |
| 1 | 402 | 11 | 0.0052 | 0.7702 | 10 | 0.0046 |
| 8 | 1008 | 9 | 0.0001 | 0.7669 | 10 | 0.0043 |
| 3 | 1297 | 13 | 0.0001 | 0.7661 | 10 | 0.0036 |
| 4 | 234 | 2 | 0.0225 | 0.7642 | 10 | 0.0042 |
| 8 | 1533 | 7 | 0.0001 | 0.763 | 10 | 0.0042 |
| 6 | 1962 | 5 | 0.0001 | 0.7628 | 10 | 0.0042 |
| 9 | 935 | 3 | 0.0022 | 0.7593 | 10 | 0.0045 |
| 9 | 1790 | 15 | 0.0001 | 0.757 | 10 | 0.0045 |
| 2 | 889 | 10 | 0.0001 | 0.7513 | 10 | 0.0036 |
| 8 | 1386 | 11 | 0.0472 | 0.7492 | 10 | 0.0039 |
| 2 | 527 | 5 | 0.0001 | 0.7488 | 10 | 0.0042 |
| 5 | 1460 | 2 | 0.0008 | 0.7398 | 10 | 0.0047 |
| 1 | 400 | 12 | 0.0001 | 0.7372 | 10 | 0.0039 |
| 10 | 692 | 12 | 0.0001 | 0.734 | 10 | 0.004 |
| 5 | 1825 | 2 | 0.0001 | 0.7275 | 10 | 0.0045 |
| 3 | 189 | 4 | 0.0001 | 0.5 | 10 | 0 |
| 7 | 63 | 6 | 0.0001 | 0.5 | 10 | 0 |

*Figure 19. First stage hyperparameter grid results –* ***static and temporal features model****. This table illustrates the performance results (rounded to 4 decimal places) for the first stage of the hyperparameter tuning procedure, ordered by performance. Each of the 20 setups is randomly generated and 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). Based on these results, the hyperparameter grid was restricted during the second hyperparameter tuning stage as follows:*

- *Number of trees: between 750 and 1400*
- *Tree depth: between 7 and 9*
- *Features allowed: between 5 and 6*
- *Learning rate: >= 0.005*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 7 | 1161 | 9 | 0.0003 | 0.7782 | 10 | 0.0043 |
| 7 | 776 | 14 | 0.0001 | 0.7774 | 10 | 0.0041 |
| 5 | 1685 | 8 | 0.0001 | 0.7764 | 10 | 0.0041 |
| 8 | 1386 | 11 | 0.0472 | 0.7739 | 10 | 0.0038 |
| 9 | 1008 | 9 | 0.0001 | 0.7717 | 10 | 0.0043 |
| 3 | 1297 | 13 | 0.0001 | 0.7689 | 10 | 0.0039 |
| 6 | 1962 | 5 | 0.0001 | 0.768 | 10 | 0.0043 |
| 9 | 1533 | 7 | 0.0001 | 0.7678 | 10 | 0.0045 |
| 10 | 1790 | 15 | 0.0001 | 0.7667 | 10 | 0.0041 |
| 3 | 527 | 5 | 0.0001 | 0.7649 | 10 | 0.0045 |
| 2 | 402 | 11 | 0.0052 | 0.7649 | 10 | 0.0038 |
| 4 | 234 | 2 | 0.0225 | 0.7625 | 10 | 0.0041 |
| 10 | 935 | 3 | 0.0022 | 0.7598 | 10 | 0.0041 |
| 2 | 889 | 10 | 0.0001 | 0.7582 | 10 | 0.0041 |
| 1 | 400 | 12 | 0.0001 | 0.7519 | 10 | 0.0042 |
| 11 | 692 | 12 | 0.0001 | 0.7507 | 10 | 0.0042 |
| 5 | 1460 | 2 | 0.0008 | 0.746 | 10 | 0.0051 |
| 6 | 1825 | 2 | 0.0001 | 0.732 | 10 | 0.0051 |
| 4 | 189 | 4 | 0.0001 | 0.5 | 10 | 0 |
| 8 | 63 | 6 | 0.0001 | 0.5 | 10 | 0 |

*Figure 20. First stage hyperparameter grid results – **static and sequential features model**. This table illustrates the performance results (rounded to 4 decimal places) for the first stage of the hyperparameter tuning procedure, ordered by performance. Each of the 20 setups is randomly generated and 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). Based on these results, the hyperparameter grid was restricted during the second hyperparameter tuning stage as follows:*

- *Number of trees: between 1000 and 1400*
- *Tree depth: between 7 and 9*
- *Features allowed: between 5 and 7*
- *Learning rate: >= 0.005*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 5 | 1685 | 8 | 0.0001 | 0.7475 | 10 | 0.0043 |
| 6 | 776 | 14 | 0.0001 | 0.7461 | 10 | 0.0038 |
| 7 | 1161 | 9 | 0.0003 | 0.7447 | 10 | 0.004 |
| 4 | 234 | 2 | 0.0225 | 0.7443 | 10 | 0.0042 |
| 1 | 402 | 11 | 0.0052 | 0.7408 | 10 | 0.0048 |
| 3 | 1297 | 13 | 0.0001 | 0.7406 | 10 | 0.0039 |
| 9 | 935 | 3 | 0.0022 | 0.7391 | 10 | 0.0039 |
| 6 | 1962 | 5 | 0.0001 | 0.7385 | 10 | 0.004 |
| 8 | 1008 | 9 | 0.0001 | 0.7373 | 10 | 0.0041 |
| 8 | 1533 | 7 | 0.0001 | 0.7354 | 10 | 0.0038 |
| 2 | 527 | 5 | 0.0001 | 0.735 | 10 | 0.0044 |
| 2 | 889 | 10 | 0.0001 | 0.7327 | 10 | 0.0041 |
| 9 | 1790 | 15 | 0.0001 | 0.7268 | 10 | 0.0036 |
| 5 | 1460 | 2 | 0.0008 | 0.7267 | 10 | 0.0042 |
| 8 | 1386 | 11 | 0.0472 | 0.7206 | 10 | 0.0046 |
| 5 | 1825 | 2 | 0.0001 | 0.7124 | 10 | 0.005 |
| 1 | 400 | 12 | 0.0001 | 0.7062 | 10 | 0.0043 |
| 10 | 692 | 12 | 0.0001 | 0.7029 | 10 | 0.0041 |
| 3 | 189 | 4 | 0.0001 | 0.5 | 10 | 0 |
| 7 | 63 | 6 | 0.0001 | 0.5 | 10 | 0 |

*Figure 21. First stage hyperparameter grid results –* ***temporal and sequential features model****. This table illustrates the performance results (rounded to 4 decimal places) for the first stage of the hyperparameter tuning procedure, ordered by performance. Each of the 20 setups is randomly generated and 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). Based on these results, the hyperparameter grid was restricted during the second hyperparameter tuning stage as follows:*

- *Number of trees: between 750 and 1400*
- *Tree depth: between 7 and 9*
- *Features allowed: between 5 and 7*
- *Learning rate: >= 0.005*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 10 | 1161 | 9 | 0.0003 | 0.7833 | 10 | 0.0039 |
| 7 | 1685 | 8 | 0.0001 | 0.7832 | 10 | 0.0038 |
| 9 | 776 | 14 | 0.0001 | 0.7831 | 10 | 0.0038 |
| 4 | 1297 | 13 | 0.0001 | 0.7790 | 10 | 0.004 |
| 12 | 1008 | 9 | 0.0001 | 0.7750 | 10 | 0.0041 |
| 11 | 1386 | 11 | 0.0472 | 0.7742 | 10 | 0.004 |
| 6 | 234 | 2 | 0.0225 | 0.7739 | 10 | 0.004 |
| 9 | 1962 | 5 | 0.0001 | 0.7725 | 10 | 0.004 |
| 12 | 1533 | 7 | 0.0001 | 0.7724 | 10 | 0.004 |
| 3 | 527 | 5 | 0.0001 | 0.7708 | 10 | 0.0043 |
| 13 | 935 | 3 | 0.0022 | 0.7702 | 10 | 0.0042 |
| 2 | 402 | 11 | 0.0052 | 0.7671 | 10 | 0.0036 |
| 2 | 889 | 10 | 0.0001 | 0.7620 | 10 | 0.004 |
| 14 | 1790 | 15 | 0.0001 | 0.7549 | 10 | 0.0054 |
| 7 | 1460 | 2 | 0.0008 | 0.7520 | 10 | 0.0049 |
| 1 | 400 | 12 | 0.0001 | 0.7430 | 10 | 0.0045 |
| 8 | 1825 | 2 | 0.0001 | 0.7391 | 10 | 0.0056 |
| 15 | 692 | 12 | 0.0001 | 0.7385 | 10 | 0.005 |
| 5 | 189 | 4 | 0.0001 | 0.5 | 10 | 0 |
| 11 | 63 | 6 | 0.0001 | 0.5 | 10 | 0 |

*Figure 22. First stage hyperparameter grid results –* ***static, temporal and sequential features model (all features model)****. This table illustrates the performance results (rounded to 4 decimal places) for the first stage of the hyperparameter tuning procedure, ordered by performance. Each of the 20 setups is randomly generated and 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). Based on these results, the hyperparameter grid was restricted during the second hyperparameter tuning stage as follows:*

- *Number of trees: between 750 and 1400*
- *Tree depth: between 7 and 9*
- *Features allowed: between 5 and 10*
- *Learning rate: >= 0.005*

## Appendix E: Hyperparameter tuning second stage results

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 5 | 880 | 7 | 0.0052 | 0.7601 | 10 | 0.0047 |
| 5 | 1101 | 7 | 0.0084 | 0.7586 | 10 | 0.0047 |
| 4 | 958 | 8 | 0.0062 | 0.7584 | 10 | 0.0045 |
| 6 | 844 | 7 | 0.0097 | 0.7582 | 10 | 0.0047 |
| 5 | 1104 | 7 | 0.0098 | 0.7581 | 10 | 0.0047 |
| 4 | 844 | 8 | 0.0099 | 0.7579 | 10 | 0.0044 |
| 6 | 1232 | 7 | 0.0084 | 0.7578 | 10 | 0.0048 |
| 4 | 1270 | 8 | 0.0061 | 0.7576 | 10 | 0.0044 |
| 5 | 1389 | 8 | 0.0064 | 0.7575 | 10 | 0.0047 |
| 6 | 995 | 8 | 0.0091 | 0.7574 | 10 | 0.0046 |
| 5 | 1191 | 8 | 0.0097 | 0.7567 | 10 | 0.0047 |
| 4 | 1179 | 8 | 0.0094 | 0.7566 | 10 | 0.0044 |
| 4 | 1100 | 9 | 0.0078 | 0.7565 | 10 | 0.0044 |
| 4 | 1065 | 9 | 0.0089 | 0.7563 | 10 | 0.0045 |
| 4 | 993 | 9 | 0.0089 | 0.7563 | 10 | 0.0045 |
| 4 | 1064 | 9 | 0.0092 | 0.7561 | 10 | 0.0045 |

*Figure 24. Second stage hyperparameter grid results – **static features model**. This table illustrates the performance results for the second stage of the hyperparameter tuning procedure, ordered by performance. Each of the 16 randomly generated setups limits the hyperparameters within the ranges identified during the first stage. Note that, since 4 hyperparameters were tuned, each hyperparameter is evaluated twice, holding all other hyperparameters constant. All setups are 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). The best-performing setup for the XGBoost algorithm is as follows:*

- *Number of trees: 880*
- *Tree depth: 7*
- *Features allowed: 5*
- *Learning rate: between 0.0052*
- *Minimum observations for another split: 10*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 2 | 1045 | 8 | 0.0055 | 0.6544 | 10 | 0.0066 |
| 1 | 1087 | 9 | 0.0109 | 0.653 | 10 | 0.0055 |
| 2 | 1154 | 7 | 0.0126 | 0.6512 | 10 | 0.0065 |
| 1 | 1114 | 7 | 0.0229 | 0.6501 | 10 | 0.0057 |
| 1 | 1116 | 7 | 0.0261 | 0.6488 | 10 | 0.0056 |
| 1 | 927 | 8 | 0.0271 | 0.6486 | 10 | 0.0054 |
| 2 | 1128 | 7 | 0.0197 | 0.6485 | 10 | 0.0065 |
| 1 | 1008 | 9 | 0.0202 | 0.6479 | 10 | 0.0055 |
| 1 | 1150 | 7 | 0.0298 | 0.6472 | 10 | 0.0058 |
| 1 | 1112 | 8 | 0.0254 | 0.6466 | 10 | 0.0053 |
| 1 | 930 | 8 | 0.0285 | 0.6455 | 10 | 0.0055 |
| 2 | 1153 | 7 | 0.0255 | 0.6451 | 10 | 0.0064 |
| 2 | 894 | 8 | 0.0253 | 0.6449 | 10 | 0.0068 |
| 2 | 1151 | 8 | 0.0192 | 0.6444 | 10 | 0.0066 |
| 2 | 808 | 9 | 0.0284 | 0.6428 | 10 | 0.0067 |
| 2 | 1154 | 8 | 0.0261 | 0.6423 | 10 | 0.0065 |

*Figure 25. Second stage hyperparameter grid results –**temporal features model**. This table illustrates the performance results for the second stage of the hyperparameter tuning procedure, ordered by performance. Each of the 16 randomly generated setups limits the hyperparameters within the ranges identified during the first stage. Note that, since 4 hyperparameters were tuned, each hyperparameter is evaluated twice, holding all other hyperparameters constant. All setups are 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). The best-performing setup for the XGBoost algorithm is as follows:*

- *Number of trees: 1045*
- *Tree depth: 8*
- *Features allowed: 2*
- *Learning rate: between 0.0055*
- *Minimum observations for another split: 10*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 6 | 1191 | 7 | 0.0051 | 0.7354 | 10 | 0.0038 |
| 6 | 1065 | 7 | 0.0061 | 0.7352 | 10 | 0.0038 |
| 5 | 844 | 7 | 0.0092 | 0.7352 | 10 | 0.0036 |
| 6 | 844 | 7 | 0.0084 | 0.7351 | 10 | 0.0037 |
| 5 | 1232 | 7 | 0.0064 | 0.7351 | 10 | 0.0036 |
| 5 | 958 | 7 | 0.0086 | 0.735 | 10 | 0.0036 |
| 5 | 1064 | 7 | 0.0094 | 0.7347 | 10 | 0.0036 |
| 6 | 993 | 8 | 0.0052 | 0.7342 | 10 | 0.0037 |
| 5 | 1389 | 8 | 0.0062 | 0.7338 | 10 | 0.0037 |
| 6 | 1100 | 8 | 0.0065 | 0.7336 | 10 | 0.0036 |
| 5 | 995 | 8 | 0.0097 | 0.7332 | 10 | 0.0035 |
| 6 | 1104 | 8 | 0.0091 | 0.7331 | 10 | 0.0034 |
| 6 | 1101 | 8 | 0.0092 | 0.7329 | 10 | 0.0034 |
| 5 | 1179 | 9 | 0.01 | 0.7314 | 10 | 0.0035 |
| 6 | 1270 | 9 | 0.0078 | 0.7313 | 10 | 0.0036 |
| 6 | 880 | 9 | 0.0097 | 0.7313 | 10 | 0.0036 |

*Figure 26. Second stage hyperparameter grid results –**sequential features model**. This table illustrates the performance results for the second stage of the hyperparameter tuning procedure, ordered by performance. Each of the 16 randomly generated setups limits the hyperparameters within the ranges identified during the first stage. Note that, since 4 hyperparameters were tuned, each hyperparameter is evaluated twice, holding all other hyperparameters constant. All setups are 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). The best-performing setup for the XGBoost algorithm is as follows:*

- *Number of trees: 1191*
- *Tree depth: 7*
- *Features allowed: 6*
- *Learning rate: between 0.0051*
- *Minimum observations for another split: 10*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 6 | 1191 | 7 | 0.0051 | 0.7798 | 10 | 0.0042 |
| 5 | 1232 | 7 | 0.0064 | 0.7797 | 10 | 0.0042 |
| 5 | 958 | 7 | 0.0086 | 0.7794 | 10 | 0.0041 |
| 6 | 1065 | 7 | 0.0061 | 0.7793 | 10 | 0.0043 |
| 6 | 844 | 7 | 0.0084 | 0.7792 | 10 | 0.0042 |
| 6 | 993 | 8 | 0.0052 | 0.7791 | 10 | 0.0043 |
| 5 | 844 | 7 | 0.0092 | 0.7788 | 10 | 0.0042 |
| 5 | 1064 | 7 | 0.0094 | 0.7786 | 10 | 0.0041 |
| 5 | 1389 | 8 | 0.0062 | 0.7784 | 10 | 0.0042 |
| 6 | 1100 | 8 | 0.0065 | 0.7781 | 10 | 0.0043 |
| 5 | 995 | 8 | 0.0097 | 0.7771 | 10 | 0.0042 |
| 6 | 1101 | 8 | 0.0092 | 0.7768 | 10 | 0.0043 |
| 6 | 880 | 9 | 0.0097 | 0.7765 | 10 | 0.0043 |
| 6 | 1104 | 8 | 0.0091 | 0.7761 | 10 | 0.0042 |
| 6 | 1270 | 9 | 0.0078 | 0.7755 | 10 | 0.0043 |
| 5 | 1179 | 9 | 0.01 | 0.7753 | 10 | 0.0042 |

*Figure 27. Second stage hyperparameter grid results –* ***static and temporal features model****. This table illustrates the performance results for the second stage of the hyperparameter tuning procedure, ordered by performance. Each of the 16 randomly generated setups limits the hyperparameters within the ranges identified during the first stage. Note that, since 4 hyperparameters were tuned, each hyperparameter is evaluated twice, holding all other hyperparameters constant. All setups are 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). The best-performing setup for the XGBoost algorithm is as follows:*

- *Number of trees: 1191*
- *Tree depth: 7*
- *Features allowed: 6*
- *Learning rate: between 0.0051*
- *Minimum observations for another split: 10*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 7 | 1328 | 7 | 0.0086 | 0.7907 | 10 | 0.0039 |
| 6 | 1351 | 7 | 0.0084 | 0.7907 | 10 | 0.0039 |
| 7 | 1208 | 7 | 0.0094 | 0.7906 | 10 | 0.004 |
| 7 | 1354 | 7 | 0.0098 | 0.7903 | 10 | 0.0039 |
| 6 | 1130 | 7 | 0.0064 | 0.7899 | 10 | 0.004 |
| 7 | 1312 | 7 | 0.0051 | 0.7897 | 10 | 0.0039 |
| 6 | 1127 | 8 | 0.0091 | 0.7897 | 10 | 0.004 |
| 7 | 1354 | 8 | 0.0092 | 0.7896 | 10 | 0.0039 |
| 6 | 1316 | 8 | 0.0062 | 0.7896 | 10 | 0.0041 |
| 6 | 1008 | 7 | 0.0061 | 0.7896 | 10 | 0.004 |
| 5 | 1287 | 8 | 0.0052 | 0.789 | 10 | 0.0041 |
| 5 | 1350 | 8 | 0.0097 | 0.7885 | 10 | 0.0039 |
| 5 | 1094 | 8 | 0.0065 | 0.7884 | 10 | 0.0041 |
| 5 | 1353 | 9 | 0.0084 | 0.7882 | 10 | 0.0039 |
| 5 | 1245 | 9 | 0.0078 | 0.7881 | 10 | 0.004 |
| 5 | 1314 | 9 | 0.0097 | 0.7874 | 10 | 0.0038 |

*Figure 28. Second stage hyperparameter grid results –* ***static and sequential features model****. This table illustrates the performance results for the second stage of the hyperparameter tuning procedure, ordered by performance. Each of the 16 randomly generated setups limits the hyperparameters within the ranges identified during the first stage. Note that, since 4 hyperparameters were tuned, each hyperparameter is evaluated twice, holding all other hyperparameters constant. All setups are 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). The best-performing setup for the XGBoost algorithm is as follows:*

- *Number of trees: 1328*
- *Tree depth: 7*
- *Features allowed: 7*
- *Learning rate: between 0.0086*
- *Minimum observations for another split: 10*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 6 | 880 | 7 | 0.0052 | 0.7528 | 10 | 0.0044 |
| 7 | 844 | 7 | 0.0097 | 0.7519 | 10 | 0.0045 |
| 6 | 1101 | 7 | 0.0084 | 0.7512 | 10 | 0.0045 |
| 5 | 958 | 8 | 0.0062 | 0.7508 | 10 | 0.0046 |
| 5 | 1270 | 8 | 0.0061 | 0.7505 | 10 | 0.0045 |
| 7 | 1232 | 7 | 0.0084 | 0.7504 | 10 | 0.0046 |
| 5 | 844 | 8 | 0.0099 | 0.7504 | 10 | 0.0046 |
| 6 | 1389 | 8 | 0.0064 | 0.7503 | 10 | 0.0048 |
| 6 | 1104 | 7 | 0.0098 | 0.7501 | 10 | 0.0046 |
| 7 | 995 | 8 | 0.0091 | 0.7499 | 10 | 0.0048 |
| 5 | 993 | 9 | 0.0089 | 0.7482 | 10 | 0.0047 |
| 5 | 1179 | 8 | 0.0094 | 0.7481 | 10 | 0.0046 |
| 5 | 1065 | 9 | 0.0089 | 0.748 | 10 | 0.0046 |
| 5 | 1100 | 9 | 0.0078 | 0.7477 | 10 | 0.0045 |
| 6 | 1191 | 8 | 0.0097 | 0.7473 | 10 | 0.0047 |
| 5 | 1064 | 9 | 0.0092 | 0.7462 | 10 | 0.0047 |

*Figure 29. Second stage hyperparameter grid results –* ***temporal and sequential features model****. This table illustrates the performance results for the second stage of the hyperparameter tuning procedure, ordered by performance. Each of the 16 randomly generated setups limits the hyperparameters within the ranges identified during the first stage. Note that, since 4 hyperparameters were tuned, each hyperparameter is evaluated twice, holding all other hyperparameters constant. All setups are 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). The best-performing setup for the XGBoost algorithm is as follows:*

- *Number of trees: 880*
- *Tree depth: 7*
- *Features allowed: 6*
- *Learning rate: between 0.0052*
- *Minimum observations for another split: 10*

| Features allowed | Trees | Tree depth | Learn rate | Mean AUC | n | Std. error |
|---|---|---|---|---|---|---|
| 9 | 1400 | 5 | 0.0116 | 0.7987 | 10 | 0.0039 |
| 7 | 1400 | 5 | 0.0116 | 0.7986 | 10 | 0.0039 |
| 7 | 750 | 5 | 0.0116 | 0.7973 | 10 | 0.0037 |
| 9 | 750 | 5 | 0.0116 | 0.7971 | 10 | 0.0038 |
| 9 | 750 | 10 | 0.0116 | 0.796 | 10 | 0.0041 |
| 7 | 750 | 10 | 0.0116 | 0.7954 | 10 | 0.004 |
| 9 | 1400 | 10 | 0.0116 | 0.7945 | 10 | 0.004 |
| 7 | 1400 | 10 | 0.0116 | 0.7936 | 10 | 0.0039 |
| 9 | 750 | 5 | 0.0715 | 0.7917 | 10 | 0.0037 |
| 7 | 750 | 5 | 0.0715 | 0.7915 | 10 | 0.0038 |
| 7 | 1400 | 5 | 0.0715 | 0.7842 | 10 | 0.0037 |
| 9 | 1400 | 5 | 0.0715 | 0.7838 | 10 | 0.0036 |
| 7 | 750 | 10 | 0.0715 | 0.7826 | 10 | 0.0036 |
| 9 | 750 | 10 | 0.0715 | 0.7816 | 10 | 0.0034 |
| 9 | 1400 | 10 | 0.0715 | 0.7757 | 10 | 0.0034 |
| 7 | 1400 | 10 | 0.0715 | 0.7754 | 10 | 0.0037 |

*Figure 30. Second stage hyperparameter grid results –* ***static, temporal and sequential features model (all features model)****. This table illustrates the performance results for the second stage of the hyperparameter tuning procedure, ordered by performance. Each of the 16 randomly generated setups limits the hyperparameters within the ranges identified during the first stage. Note that, since 4 hyperparameters were tuned, each hyperparameter is evaluated twice, holding all other hyperparameters constant. All setups are 10-fold-cross-validated on the downsampled training dataset (12304 observed sessions). The performance metric is the area under the curve (AUC). The best-performing setup for the XGBoost algorithm is as follows:*

- *Number of trees: 1400*
- *Tree depth: 5*
- *Features allowed: 9*
- *Learning rate: between 0.0116*
- *Minimum observations for another split: 10*