Expert Systems With Applications 150 (2020) 113342

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Predicting online shopping behaviour from clickstream data using deep learning

Dennis Koehn[a], Stefan Lessmann[a,*], Markus Schaal[b]

[a] Chair of Information Systems, Humboldt University of Berlin, Spandauer Str. 1, Berlin 10178, Germany
[b] Webtrekk GmbH, Robert-Koch-Platz 4, Berlin 10115, Germany

ARTICLE INFO

Article history:
Received 18 December 2018
Revised 4 February 2020
Accepted 25 February 2020
Available online 26 February 2020

Keywords:
Deep learning
e-commerce
Recurrent neural networks
Digital marketing

ABSTRACT

Clickstream data is an important source to enhance user experience and pursue business objectives in e-commerce. The paper uses clickstream data to predict online shopping behavior and target marketing interventions in real-time. Such AI-driven targeting has proven to save huge amounts of marketing costs and raise shop revenue. Previous user behavior prediction models rely on supervised machine learning (SML). Conceptually, SML is less suitable because it cannot account for the sequential structure of clickstream data. The paper proposes a methodology capable of unlocking the full potential of clickstream data using the framework of recurrent neural networks (RNNs). An empirical evaluation based on real-world e-commerce data systematically assesses multiple RNN classifiers and compares them to SML benchmarks. To this end, the paper proposes an approach to measure the revenue impact of a targeting model. Estimates of revenue impact together with results of standard classifier performance metrics evidence the viability of RNN-based clickstream modeling and guide employing deep recurrent learners for campaign targeting. Given that the empirical analysis shows RNN-based and conventional classifiers to capture different patterns in clickstream data, a specific recommendation is to combine sequence and conventional classifiers in an ensemble. The paper shows such an ensemble to consistently outperform the alternative models considered in the study.

1. Introduction

The ability to personalize marketing communication and service offerings is a major benefit of digital marketing and e-commerce. Every interaction with a firm's website leaves a digital footprint that provides information related to customer interests, preferences, and the context in which they have visited the website. Through analyzing corresponding data, marketers can actively manage visitor interactions, for example through dynamically changing website layout and content and incorporating marketing stimuli such as sign-up buttons or digital coupons (Radcliffe & Surry, 2011). Such AI-driven targeting can save huge amounts of marketing costs and raise online sales provided that the targeting model succeeds in estimating customer responsiveness accurately (Lessmann, Haupt, Coussement, & De Bock, 2019). The paper considers the targeting of digital coupons, which are a popular form of target marketing in online shopping contexts (Reimers & Xie, 2019). A company specialized in digital marketing services supports this study providing real-world e-commerce data and reports that its customers have saved six-digit numbers of marketing budget due to improved model-based targeting of e-coupons. This hints at the potential business value of sound targeting. The paper focuses on means to develop corresponding targeting models by leveraging clickstream data and supervised machine learning (SML). This focus is omnipresent in the paper. However, we wish to note that the type of algorithms that we employ for coupon targeting are equally capable of predicting other forms of online customer behavior including newsletter sign-up, ad click, product return, to name only a few.

A clickstream represents a sequence of page views describing the navigational history of a user session. This type of data has proven useful in web usage mining and to generate real-time predictions of online shopping behavior (Bucklin & Sismeiro, 2009). A common approach, which is also taken in this paper, is to learn a decision function that categorizes current website visitors into groups using past sessions with known outcomes. Corresponding models are called classifiers and belong to the family of SML algorithms. For coupon targeting, we consider two groups. Shop visitors who make a purchase in their current session are labeled as

* Corresponding author.
E-mail addresses: dkoehn@hotmail.de (D. Koehn), stefan.lessmann@hu-berlin.de (S. Lessmann), markus.schaal@webtrekk.com (M. Schaal).

https://doi.org/10.1016/j.eswa.2020.113342
0957-4174/© 2020 Elsevier Ltd. All rights reserved.

buyers. If no purchase is made in that session, we label the visitor as non-buyer. Clearly, issuing a coupon to an actual buyer implies lowering sales margin and should be avoided.

Clickstream data occurs in a sequential form whereas standard SML algorithms employ tabular data. It is not obvious how to use these algorithms for user behavior prediction from clickstream data. The prevailing approach in prior work, called *clipping at every click*, is to perform training and prediction at the level of an individual page view (VanderMeer, Dutta, Datta, Ramamritham, & Navanthe, 2000). Treating each page view of a session as a single instance facilitates the use of SML at the cost of breaking the sequential structure of the user clickstream. The paper takes a different path and approaches the user behavior prediction task as a sequence classification problem. We feed user sessions as sequences of page views to an algorithm that directly predicts the session outcomes. To achieve this, the paper explores a family of deep learning algorithms called recurrent neural networks (RNNs) that were specifically introduced for sequential data processing (Goodfellow, Bengio, Courville, & Bengio, 2016). We posit that RNNs represent a more suitable approach to capture the sequential nature of clickstream data and overcome several conceptual flaws of previous attempts to derive user behavior predictions from clickstream data using SML. Section 2 elaborates on this proposition.

Given that analytical models for personalizing and targeting marketing communication are of crucial importance for e-tailers (Gubela et al., 2017), the hypothesized superiority of RNNs for clickstream classification motivates us to test the adequacy of RNNs for clickstream classification in a real-world e-commerce application. The task of targeting online shoppers with digital coupons serves as a testbed in which we systematically compare different types of RNNs to established standard SML algorithms. This setting allows us to make three contributions to literature. A first contribution consists of establishing the conceptual differences between alternative clickstream modeling approaches and identifying their respective merits and demerits. This distinction is important because prior literature concentrates on either sequence classification or standard SML algorithms but does not consider them in a unified framework. As a consequence, a comprehensive picture of the alternative options to turn clickstream data into actionable user behavior predictions is lacking and provided here.

The paper also contributes to the empirical literature on SML for marketing decision support and e-commerce analytics. We employ different types of RNNs for clickstream data modeling and real-time classification of e-shop visitors into buyers and non-buyers. We also benchmark RNNs against established classification models. The results of our study augment the conceptual comparison of alternative clickstream modeling frameworks with empirical evidence of their effectiveness using real-world e-commerce data.

The third and last contribution of the paper comes from the development and application of a novel framework to estimate the business value of a coupon targeting model. Given the wide use of e-coupons as a digital marketing instrument (Gubela, Lessmann, & Jaroszewicz, 2019), insight into the monetary value of a targeting model is especially valuable for corporate practice and can inform decisions in the scope of model maintenance such as whether to revise a deployed model or replace it with a challenger. Applying the new evaluation approach to our empirical results also expands the scope of the comparative study and assesses alternative clickstream-based targeting models from a value perspective.

2. Conversion classification using clickstream data

The section introduces the conversion classification task and the notation we use throughout the paper. Subsequently, we elaborate on different modeling approaches for clickstream classification.

2.1. Problem setting

We define conversion classification as the task to predict whether a website visitor performs some action that a marketer strives to trigger. Without loss of generality, we consider a transactional e-commerce website and assume the intended behavior to be a purchase. Likewise, a marketer might be interested in making a visitor leave contact data (lead generation), download an app, etc. The specific definition of what action constitutes a conversion has no methodological implications for clickstream classification.

We state the clickstream classification problem as follows. Let the *training set* consist of a collection of user sessions $\mathcal{X} = \left(\mathbf{X}_1, \ldots, \mathbf{X}_N\right)$ and their known binary outcomes or labels $\mathbf{y} = \left(y_1, \ldots, y_N\right)$. A label y_i is 0 if the i-th session leads to a conversion and 1 otherwise. Furthermore, let $\tilde{\mathcal{X}} = \left(\tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_{\tilde{N}}\right)$ denote new incoming sessions the labels of which, $\tilde{\mathbf{y}} = \left(\tilde{y}_1, \ldots, \tilde{y}_{\tilde{N}}\right)$, are unknown. Each session from $\mathcal{X}$ and $\tilde{\mathcal{X}}$ includes multiple page views. More specifically, a session i is defined by $\mathbf{X}_i = \left(\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(T_i)}\right)$, where $\mathbf{x}_i^{(t)} \in \mathbb{R}^d$ is the t-th page view of session i containing d different features describing the page view. Note that the number of page views, that is the session length T_i, varies across sessions. Consequently, $\mathbf{X}_i$ can be represented by a $T_i \times d$ matrix, where the size of the first dimension is arbitrary. Given this setting, the goal of clickstream classification is to find a function:

$$f : \mathcal{X} \to \mathbf{y}, \tag{1}$$

which can correctly predict unknown labels of new live sessions, i.e.

$$f(\tilde{\mathbf{X}}_i) = \tilde{y}_i \tag{2}$$

should hold for many of the $\tilde{N}$ incoming sessions. For most algorithms, the output of f is not a label but a probability estimate indicating the likelihood of the outcome $\tilde{y}_i = 1$. In order to obtain a discrete class prediction, we define a probability threshold τ. If the probability outcome exceeds τ, the predicted label is 1; and 0 otherwise. The specific structure of the input data gives rise a major difference compared to standard binary classification. Recall that a session $\mathbf{X}_i$ is represented by a matrix with a dynamic first dimension. Hence, the collections of sessions $\mathcal{X}$ and $\tilde{\mathcal{X}}$ can not be represented by a matrix without further assumptions. SML algorithms such as logistic regression require a matrix of fixed-sized dimensions as input. The following discussion provides an overview of strategies to cope with this issue.

2.2. Aggregation on the session level

Arguably the most straightforward solution to prepare clickstream data for supervised classification is to aggregate the features of the page views on the session level. This approach transforms the three-dimensional data structure of $\mathcal{X}$ into a matrix $\mathbf{X^a} \in \mathbb{R}^{N \times d}$. This matrix can then be used by common classifiers to learn the function f, as illustrated in Eq. (1). The necessary aggregation may be achieved by computing summary statistics of each session. This approach is taken by earlier studies on clickstream data such as Van den Poel and Buckinx (2005) or Moe and Fader (2004). The latter paper develops a topology of online visitors from an e-commerce shop and derives several aggregated features including the general session descriptors (e.g., number of page views), users' browsing focus (e.g., share of product page views), variety of products and categories viewed (e.g., average number of unique products viewed within a category) and repeated view measures (e.g., maximum number of views on the same product page). Apart from summary statistics, more sophisticated feature extraction methods have been introduced. Examples include Shapoval and Setzer (2018) who propose two unsupervised algorithms to extract session-level features or Suh, Lim, Hwang, and Kim (2004) who explore the use of association rule mining. An obvious drawback of aggregation on the session level is the loss of information. Furthermore, the effectiveness and the construction of aggregation features is data-dependent and requires domain knowledge.

2.3. Clipping at every click

An alternative approach to handle clickstream data is known as *clipping at every click* (VanderMeer et al., 2000). As we discuss below, this approach is commonly adopted in related research (Baumann, Haupt, Gebert, & Lessmann, 2018; Park & Park, 2016). Similar to the aggregation method, clipping at every click transforms the complex structure of clickstream data into the form of a standard classification problem. However, as opposed to labeling whole sessions, the transformation is realized by classifying each page view on its own. To that end, the page views of all sessions are concatenated such that the result is a two-dimensional matrix. To match the number of rows of this matrix during model training, each page view receives the class label of the session. In the specific case of conversion classification this means that all page views of a session are assigned the label 1 if the session leads to a conversion; and the label 1 otherwise. Formally, concatenating page views converts $\mathcal{X}$ into the matrix $\mathbf{X^c} = \left(\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_1^{(T_1)}, \mathbf{x}_2^{(1)}, \ldots, \mathbf{x}_N^{(T_N)}\right)$, which has $|T| = \sum_{i=1}^{N} T_i$ rows and d columns. The labels for each page view are concatenated as well resulting in a $|T|$-dimensional vector $\mathbf{y^c} = \left(y_1^{(1)}, \ldots, y_1^{(T_1)}, y_2^{(1)}, \ldots, y_N^{(T_N)}\right)$. After training the model, the learned decision function takes single page views, i.e. d-dimensional vectors, as an input for prediction. Thus, a standard classification problem is obtained that needs a matrix as an input while training and predicts the labels of new data points represented by vectors of feature values.

After the transformation, the session classification task can be approached with any binary classification algorithm. However, in contrast to the aggregation approach, clipping at every click provides the flexibility to predict outcomes at any arbitrary point during the session and is, therefore, more suitable for a live scenario. On the other hand, by treating each page view as a single data point, the representation violates the assumption of data being independently and identical (iid) distributed. This is due to the likelihood of auto-correlation between page views within the same user session. Such non-iid property is an issue for standard classifiers since they are not capable of modeling temporal dependencies and treat each data point in isolation. As a remedy, information on session context from previous page views is typically given to the classifiers by the means of engineered features. Crafting meaningful features that include all relevant information is a time-consuming process and requires detailed domain knowledge. Another often neglected problem of clipping every click concerns the evaluation of the resulting classifiers. When evaluating the predictive power on a holdout set with standard classification metrics, each page view is equally weighted. This might not be desirable for use cases such as e-coupon systems, because the ultimate decision is taken on the session level and not on the page view level.

To illustrate this problem it is helpful to look at an extreme example where the test set only consists of two sessions. Let us assume that the first session contains 99 page views and the second only one page view, i.e. $\tilde{X}_1 \in \mathbb{R}^{99\times d}$ and $\tilde{X}_2 \in \mathbb{R}^{1\times d}$. Now consider the case where the classifier predicts each label in $\tilde{X}_1$ correctly but fails to predict the right label for the page view in $\tilde{X}_2$. The classification accuracy on the page view level would be 99%, indicating that we have a good classifier. However, the decisions taken based on such a classifier in an e-coupon setting would be wrong half of the time. A possible remedy to this problem is to evaluate the classifier on the session-level. Nevertheless, it remains somewhat unclear how to construct a global session prediction out of page view prediction outputs. Should we take the last, the maximum or the in case of probability outputs the average predicted output belonging to a session?

In summary, the shortcomings of the clipping at every click approach come from the fact that it ignores the original structure of clickstream data, which can be seen as a collection of page view sequences. Hence, a more natural approach would be to model sessions as sequences and predict their outcome directly. This approach will be discussed in the following section.

2.4. Sequence classification

Sequence classification is the task of labeling sequences (Graves, 2012). This task has been used to tackle problems in a broad range of domains (see Xing, Pei, and Keogh (2010) for an overview of applications). In the context of clickstream data, we will model sessions as sequences of page views. In particular, we will treat each session as a single instance that has a single label. Furthermore, we take the temporal order of the page views $\mathbf{X}_i = \left(\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(T_i)}\right)$ in the session, marked by the upper subscripts, into account.

The advantages of sequence classification for the task are threefold. First, the approach allows us to directly predict the session label. This simplifies the evaluation of the classifier and models the decision problem of an e-coupon system in a more direct fashion. This is because the level of the classifier outcome matches the level of the actual decision problem. Second, by modeling sessions as sequences, we avoid the problem of page views within the same session violating the iid assumption. More specifically, sequence classification captures the temporal ordering of page views and the resulting dependencies explicitly. A page view $\mathbf{x}_i^{(t)}$ is allowed to depend on all previous page views of its session $\mathbf{x}_i^{(t-1)}, \ldots, \mathbf{x}_i^{(1)}$. Last, predicting the outcomes of incomplete sessions allows us to include additional features that can only be measured after a particular page view has ended. For instance, Bogina and Kuflik (2017) argue that the time spent on a page of a specific item might reveal the interest of a user for that item. Following this argumentation, it would be useful to include a feature that measures the time a user has spent on a page view including the final page view before predicting. Such information is accessible in a live setting when performing sequence classification because the incoming inputs are incomplete sequences consisting of page views that have already ended at the time the prediction is made. In contrast to this, when predicting single page views in a live scenario, it is only feasible to use information up to the page view that precedes the page view for which a prediction is sought. Consequently, a feature such as the time spent on a specific page view can not be included in this case.

A challenge of sequence classification is that it requires the use of a learning algorithm that can handle the complex input data structure. Recall that the input of f from Eq. (1) is a collection of sessions denoted by $\mathcal{X}$. This input has three dimensions. The first dimension represents the sessions, the second dimension represents the page views within sessions, and the third dimension represents the features. Hence, the input data is a three-dimensional array or third-order tensor, i.e. $\mathcal{X} \in \mathbb{R}^{N\times T_i\times d}$. Since the size of the second dimension differs for each session, an additional requirement for a potential classifier is introduced. The classifier requires the ability to handle sequences of variable sizes. Clearly, without major modification, well-established classifiers such as logistic regression or random forest can not fulfill these requirements. RNN based architectures, on the other hand, provide a solution. After reviewing the literature, we will discuss their suitability for sequence classification in Section 4.

3. Related work

The section presents a systematic review of related research. We focus on studies that process clickstream data using supervised learning in the context of marketing use cases. Corresponding studies either build models to improve product recommendations or to implement an e-coupon system. An overview of related work is given in Table 1, where we also include the focal paper for the sake of comparison. Table 1 reports the modeling steps taken by each study in four categories. These categories include the target variable that is used (Target), whether the considered models are evaluated based on their business value for the focused application (Business Value) and whether sessions are modeled as sequences (Sequence Labeling). Furthermore, the table contains a column Live Scenario that indicates if the taken approach facilitates real-time predictions. This is not the case for all of the studies. For example, Moe and Fader (2004) and Park and Park (2016) build models to predict conversions for the next visit of the user rather than for the current live session.

Table 1 reveals that the majority of previous studies focus on predicting conversion. Recall that conversion can be defined as any preferable action of the user. For example, Chan et al. (2014) build a conversion model with the data of a car retailer website. They define a conversion as the user completing a form to contact a specific car dealer. Aside from conversion, prior literature also includes product choice, next page and last page as targets. These prediction models are motivated by improving product recommendations for the user.

Nearly half of the studies perform aggregation on the session level or use clipping at every click. Table 1 identifies corresponding studies with a cross in the category *sequence labeling*. Following the terminology of Graves (2012), we use this more general term to include all studies that handle sequences and model the temporal dependencies of page views through the means of a prediction model regardless whether the model's output is a single session label, i.e. *sequence classification*, or multiple labels for each page view, i.e. *segment classification*. If a paper falls in this category, the concrete model name or model class is reported. Within this category, earlier research focuses on Markov models or develops customized probabilistic approaches, e.g. Montgomery, Li, Srinivasan, and Liechty (2004) or Sismeiro and Bucklin (2004). In contrast to this, more recent studies often consider different kinds of RNNs.

Inspired by the success in natural language processing (NLP), a few recent studies try to learn ontologies of product categories from clickstream data. To achieve this, Arora and Warrier (2016) learn vector representations for clusters of products using the skip-gram model commonly referred to as *word2vec* (Mikolov, Chen, Corrado, & Dean, 2013). Using an analogy of documents and words from NLP applications, product vectors are learned by a *word2vec* model using sessions as documents and product attributes in the sessions as words. As a result, products that are often searched successively have a similar product vector. This approach was further adopted by Tamhane, Arora, and Warrier (2017) who feed the learned product vectors to a gated recurrent unit (GRU) to predict the product group of the sessions last page view.

To the best of our knowledge, the only studies that consider RNNs for conversion classification are Wu, Tan, Duan, Liu, and Mong Goh (2015) and Toth, Tan, Di Fabbrizio, and Datta (2017). The former study employs a bidirectional RNN architecture to clickstream data. In addition to a recurrent feedback loop, which memorizes behavior in a session from past page views, a forward loop that captures behavior in the following page views is added. Although this is an innovative idea, bidirectional RNNs are effectively not applicable in a live prediction scenario as information from future page views is needed to predict the session's outcome at the current page view. Toth et al. (2017) classify sessions into *purchase, abandoned basket*, and *browse only* using a mixture of *long short-term memory models* (LSTMs). In particular, three LSTMs are trained to estimate the posterior distribution for each of the three outcomes. Compared to our work, their approach exhibits three major differences. First, in their conclusion, Toth et al. (2017) argue that future research should concentrate on single RNNs ar-

Table 1
Overview of relevant literature (in alphabetical order).

Reference	Target	Live scenario	Business value	Sequence labeling
This work	Conversion	✓	Revenue Gain	LSTM and GRU
Arora and Warrier (2016)	Last Page	✓	×	×
Barkan, Brumer, and Koenigstein (2016)	Last Page	✓	×	×
Baumann et al. (2018)	Conversion	✓	Lift	×
Baumann et al. (2019)	Conversion	✓	Revenue Gain	×
Bertsimas et al. (2003)	Conversion	✓	Lift	HMM
Bogina and Kuflik (2017)	Next Page	✓	×	GRU
Borges and Levene (2007)	Product Choice	✓	×	Markov
Chan et al. (2014)	Conversion	✓	×	Markov Model
Hidasi, Karatzoglou, Baltrunas, and Tikk (2015)	Next Page	✓	×	GRU
Iwanaga, Nishimura, Sukegawa, and Takano (2016)	Product Choice	×	×	×
Lakshminarayan, Kosuru, and Hsu (2016)	Conversion	✓	×	Markov Model
Moe (2003)	Conversion	×	×	×
Moe and Fader (2004)	Conversion	×	×	×
Montgomery et al. (2004)	Conversion	✓	×	Dynamic Probit Model
Olbrich and Holsing (2011)	Conversion	×	×	×
Park and Chung (2009)	Conversion	×	×	×
Park and Park (2016)	Conversion	×	Lift	HMM
Sismeiro and Bucklin (2004)	Conversion	✓	Lift	Probabilistic Model
Suh et al. (2004)	Conversion	✓	Lift	×
Tamhane et al. (2017)	Last Page	✓	×	GRU
Toth et al. (2017)	Conversion	✓	×	LSTM Mixture Model
Van den Poel and Buckinx (2005)	Conversion	×	×	×
Vieira (2015)	Conversion	×	×	×
Wu, Chiu, and Lin (2005)	Conversion	✓	×	HMM
Wu et al. (2015)	Conversion	×	×	Bidirectional LSTM
Zhao et al. (2016)	Product Choice	×	×	×

chitectures rather than on a mixture of RNNs, which is an advice that we follow in this work. Second, their study compares the proposed model with a rarely applied variant of a hidden Markov model (HMM) developed by Bertsimas, Mersereau, and Patel (2003), while this work includes four well-established learning algorithms for benchmarking. Given the popularity of supervised learners in the field (see Table 1), benchmarking against industry standards is crucial to establish the merit of novel technology from a practitioner perspective. Last, Toth et al. (2017) report standard classification metrics such as precision, recall, and f1-score but do not assess their model based on its monetary value.

4. Recurrent neural networks for clickstream data

The section briefly introduces RNNs, reviews the specific RNN architectures employed in the paper, and elaborates on their suitability for clickstream data. We refer readers to Goodfellow et al. (2016) for a fully-comprehensive coverage of these and other deep learning models. Note that our notation follows Lipton, Berkowitz, and Elkan (2015).

4.1. RNN fundamentals

Simple feed-forward neural networks such as *multi-layer perceptron* (MLP) deliver a flexible and widely applicable modeling framework. Nevertheless, being restricted to forward connections, MLPs cannot capture temporal dependencies in sequential data. RNNs were introduced to model sequential data more appropriately. To achieve this, RNNs incorporate recurrent layers, which extend feed-forward layers in that they contain forward connections to all units of the following layer and also cycles that feed the output of an unit back into that unit. Cyclical, or recurrent, connections facilitate including ordering information into the network (Rojas, 1996). In the case of clickstream data, a recurrent layer is useful to capture navigational patterns that happen at different times during the session. Navigational patterns of a session i are encoded into feature values of the page view vectors $\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(T_i)}$. In a live scenario, we want to predict the outcome of a session at time $t < T_i$. The hidden states of a recurrent layer at this time step can be computed as:

$$\mathbf{h}^{(t)} = g\left(\mathbf{W}^{\mathbf{hx}}\mathbf{x}_i^{(t)} + \mathbf{W}^{\mathbf{hh}}\mathbf{h}^{(t-1)} + \mathbf{b_h}\right), \tag{3}$$

where $\mathbf{W}^{\mathbf{hx}}$ and $\mathbf{W}^{\mathbf{hh}}$ are the matrices containing the connection weights from input to hidden layer and hidden layer to itself in the next time step, respectively. $\mathbf{b_h}$ is the bias vector of the hidden layer. Furthermore, g is a non-linear function referred to as *activation*. A common choice for g is either the *tanh*, i.e $g(z) = \frac{exp(z)-\exp(-z)}{exp(z)+\exp(-z)}$ or *relu*, i.e. $g(z) = \max(0, z)$. The hidden states play a crucial role in obtaining desirable properties of RNNs for sequences. First, they incrementally add information from each time step to the network and thus model temporal dependencies. Second, the parameters of the hidden state are shared across all sessions and time steps. This parameter sharing makes it possible to learn from inputs of different shapes (Goodfellow et al., 2016); that is sessions with a different number of page views in our setting. Given $\mathbf{h}^{(t)}$, the network can output predictions by

$$\hat{y}_i^{(t)} = \sigma\left(\mathbf{W}^{\mathbf{yh}}\mathbf{h}^{(t)} + b_y\right), \tag{4}$$

where $\mathbf{W}^{\mathbf{yh}}$ is the connection weight matrix from the hidden to the output layer and b_y is the output layer bias. Since conversion classification is a binary classification problem, the network output is passed through a *sigmoid* function, i.e. $\sigma(z) = 1/1 + \exp(-z)$, to ensure that outputs are between 0 and 1.

In theory, RNNs are able to model contextual dependencies over an infinite time horizon (Mikolov, Karafiát, Burget, Černocký, & Khudanpur, 2010). However, in practice they have serious difficulties to learn long-term dependencies as the values of the gradient for many time steps back often become marginally small or explodes (Bengio, Simard, & Frasconi, 1994). This phenomenon known as the *vanishing gradient problem* provides the motivation for the two architectures discussed next.

4.2. Long short-term memory model

As a remedy to the vanishing gradient problem, Hochreiter and Schmidhuber (1997) proposed the *long-short term memory model* (LSTM). Instead of ordinary recurrent units, the nodes of an LSTM layer are defined by memory cells illustrated in Fig. 1. A memory cell consists of four parts, an input gate $i_c^{(t)}$, an input unit with a self-recurrent cycle $g_c^{(t)}$, a forget gate $f_c^{(t)}$ and an output gate $o_i^{(t)}$. In a similar fashion as in Eq. (3), the input vector $\mathbf{x}_i^{(t)}$ and the hidden state of the previous layer $\mathbf{h}^{(t-1)}$ are processed through each of these components and are transformed into linear combinations that are wrapped by a non-linear activation.

The values of the input gate and the input unit of all memory cells in an LSTM layer are given by

$$\mathbf{i}^{(t)} = \left(\mathbf{W}^{\mathbf{ix}}\mathbf{x}_i^{(t)} + \mathbf{W}^{\mathbf{ih}}\mathbf{h}^{(t-1)} + \mathbf{b_i}\right). \tag{5}$$

$$\mathbf{g}^{(t)} = tanh\left(\mathbf{W}^{\mathbf{gx}}\mathbf{x}_i^{(t)} + \mathbf{W}^{\mathbf{gh}}\mathbf{h}^{(t-1)} + \mathbf{b_g}\right) \tag{6}$$

Note that we use the to-from notation for the connection weight matrices from Lipton et al. (2015) here, i.e. $\mathbf{W}^{\mathbf{ix}}$ is the connection weight matrix of the input values to the input gate, $\mathbf{W}^{\mathbf{ih}}$ the connection weight matrix of the hidden state to the input gate and so on. Intuitively, $\mathbf{i}^{(t)}$ and $\mathbf{g}^{(t)}$ determine which new information of the current input is used to update the cell state. The cell state, represented by the middle node in Fig. 1, acts as a long-term memory as it stores signals from all past time steps. In order to exclude outdated information from the cell state, Gers, Schmidhuber, and Cummins (1999) introduced a forget gate, i.e.

$$\mathbf{f}^{(t)} = \sigma\left(\mathbf{W}^{\mathbf{fx}}\mathbf{x}_i^{(t)} + \mathbf{W}^{\mathbf{fh}}\mathbf{h}^{(t-1)} + \mathbf{b_f}\right). \tag{7}$$

Given these three parts of the memory cell, the values of the cell state can be computed. Let $\odot$ denote element-wise multiplication,

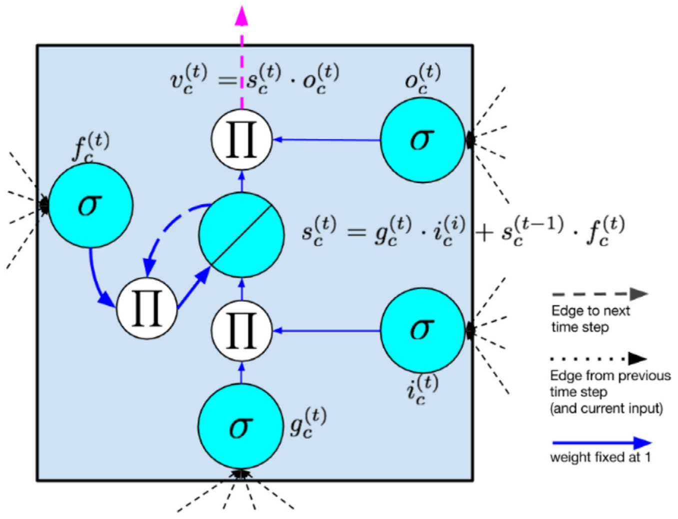
$v_c^{(t)} = s_c^{(t)} \cdot o_c^{(t)}$
$o_c^{(t)}$
$f_c^{(t)}$
$s_c^{(t)} = g_c^{(t)} \cdot i_c^{(i)} + s_c^{(t-1)} \cdot f_c^{(t)}$
$i_c^{(t)}$
$g_c^{(t)}$
σ
∏
Edge to next time step
Edge from previous time step (and current input)
weight fixed at 1

Fig. 1. LSTM memory cell taken from Lipton et al. (2015).

then the cell state is defined as:

$$\mathbf{s}^{(t)} = \mathbf{g}^{(t)} \odot \mathbf{i}^{(t)} + \mathbf{s}^{(t-1)} \odot \mathbf{f}^{(t)}. \tag{8}$$

Because the cell state includes signals from the present and the past, not all of these signals might be relevant to the specific current output of the cell. Despite this, it might be worth not to throw away all irrelevant information as some of them might become relevant again when predicting outcomes at later time steps. For example, consider the case in which a user has navigated through two different product categories such as shoes and t-shirts. Further assume that the user has looked at shoes in the last few page views. On the one hand, it would be desirable to keep the information from t-shirt related page views in case the user resumes the search for this product category in the future. Hence, this information should not be deleted from the cell state. However, when predicting the outcome of the session at the current page view, these signals might be negligible while patterns found at previous page views involving shoes are more relevant. For this reason, the LSTM does not simply override the cells state but employs an output gate:

$$\mathbf{o}^{(t)} = \sigma\left(\mathbf{W}^{\mathbf{ox}}\mathbf{x}_i^{(t)} + \mathbf{W}^{\mathbf{oh}}\mathbf{h}^{(t-1)} + \mathbf{b_o}\right) \tag{9}$$

that ensures that only currently relevant information of the past and the present are outputted while passing the complete cell state on to the next layer. As a result, a version of cell state that is filtered by $\mathbf{o}^{(t)}$ is used as the network output, i.e.

$$\mathbf{h}^{(t)} = tanh\left(\mathbf{s}^{(t)}\right) \odot \mathbf{o}^{(t)}. \tag{10}$$

The final network prediction emerges from forwarding $\mathbf{h}^{(t)}$ to an output layer given in Eq. (4).

4.3. Gated recurrent unit

The discussion LSTM cells illustrates that learning long term dependencies comes with the cost of several additional model parameters. It would be preferable to simplify the LSTM such that fewer model parameters are necessary without losing its key benefits. This is the promise of the gated recurrent unit (GRU) introduced by Cho, Van Merriënboer, Bahdanau, and Bengio (2014). Unlike the LSTM, the GRU has no separate cell state that is passed from layer to layer. Instead, the GRU cell is able to adaptively remember and forget signals from hidden states of past layers. To that end, a GRU cell contains a reset gate and an update gate, which are defined by:

$$\mathbf{r}^{(t)} = \sigma\left(\mathbf{W}^{\mathbf{rx}}\mathbf{x}_i^{(t)} + \mathbf{W}^{\mathbf{rh}}\mathbf{h}^{(t-1)} + \mathbf{b_r}\right) \tag{11}$$

$$\mathbf{u}^{(t)} = \sigma\left(\mathbf{W}^{\mathbf{ux}}\mathbf{x}_i^{(t)} + \mathbf{W}^{\mathbf{uh}}\mathbf{h}^{(t-1)} + \mathbf{b_u}\right). \tag{12}$$

While the former is used to generate candidate cell outputs:

$$\tilde{\mathbf{h}}^{(t)} = tanh\left(\mathbf{W}^{\tilde{\mathbf{h}}\mathbf{x}}\mathbf{x}_i^{(t)} + \mathbf{r}^{(t)} \odot \mathbf{W}^{\mathbf{rh}}\mathbf{h}^{(\mathbf{t-1})} + \mathbf{b}_{\tilde{\mathbf{h}}}\right), \tag{13}$$

the latter combines the characteristics of the input gate and the forget gate from the LSTM into one single gate. The final cell output is computed from a linear interpolation between the output of the previous layer and the generated candidate output using the values from the update gate for the nodes in the corresponding GRU layer $u_c^{(t)}$ as weights:

$$h_c^{(t)} = u_c^{(t)}\tilde{h}_c^{(t)} + (1 - u_c^{(t)})h_c^{(t-1)}. \tag{14}$$

Hence, the update gate controls how much of the previous memory content is to be forgotten and how much of the new memory content is to be added (Chung, Gulcehre, Cho, & Bengio, 2015).

As the GRU contains fewer gates than the LSTM, it needs fewer computations to update a unit. Since both approaches share many properties, the choice of which to use on a specific problem cannot be generally answered beforehand (Chung, Gulcehre, Cho, & Bengio, 2014). Therefore, we adapt both variants in the proposed architectures presented next.

4.4. Proposed RNN architectures for clickstream classification

To test their usefulness for conversion classification, we develop neural network architectures for sequence classification using respectively, LSTM and GRU cells. The layers of this architecture are illustrated in Fig. 2, in which each RNN cell acts as a placeholder for either an LSTM cell or a GRU cell. From hereon, we refer to this architecture as LSTM or GRU depending on which type of cells is used. Both variants can have a single recurrent layer or include multiple layers stack behind each other. The last recurrent layer is followed by a dropout layer (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014), which provides a computationally inexpensive but powerful method of regularizing a broad family of neural networks (Goodfellow et al., 2016). At the training phase, dropout randomly cuts off connections by setting a fraction of the incoming weights from the previous layer to zero. The fraction of the chosen weights constitutes a hyper-parameter, which we refer to as dropout rate. After regularizing the outcomes of the last recurrent layer, we include a pooling layer. Pooling aggregates the weights from time steps that are in the neighborhood of the specified kernel size. We set the kernel size to 3, which means that the weights of the groups with the time steps $t = 1, 2, 3,\ t = 4, 5, 6$ and so on are aggregated. Aggregation is either done by taking the maximum weight (*MaxPooling*) or the average weight (*AvgPooling*) of a group. Similar to the dropout_rate and the number of recurrent layers, we treat the pooling type as a hyper-parameter and let the model selection process (explained in Section 5.4) choose whether to perform *MaxPooling, AvgPooling* or no pooling at all. Pooling layers are commonly applied in the context of convolutional neural networks (CNNs) but have also been used in RNN architectures (Lee & Dernoncourt, 2016; Wang, Yu, & Miao, 2017). As a final step, the hidden states belonging to the last time steps of the processed input sequences are extracted and put into a feedforward layer described by Eq. (4). We fix the number of hidden nodes of this final layer to 128.

5. Experimental setup

5.1. Data

Our data set contains user sessions from the website of an online shop and was collected in the period May 20th to July 20th, 2018. The shop providing the data focuses on selling fashion items and wishes to stay anonymous. In order to prepare the data for analysis, we perform the following preprocessing steps. First, the data set includes 5 sessions with more than 400 page views. We consider the the large number of page views evidence that corresponding sessions have been generated by bots, and delete them from the data set. Next, we exclude all page views on the checkout page and their successors within the same session. Sessions already in the checkout process are not suitable for coupon targeting (Bertsimas et al., 2003). Furthermore, we delete the last three page views of all sessions. This is to simulate a live scenario in which a classifier needs to predict the outcome of an incomplete session. As a side effect, we also get rid of small sessions that have 3 or fewer page views. Finally, if a session still contains more than 100 page views after the previous cleaning steps, we prune it to a length of 100. Pruning helps to reduce memory consumption when training RNN-based sequence classifiers. Common deep learning software packages such as *pytorch* (Paszke et al., 2017) or *tensorflow* (Abadi et al., 2016) rely on fixed size data structures. In

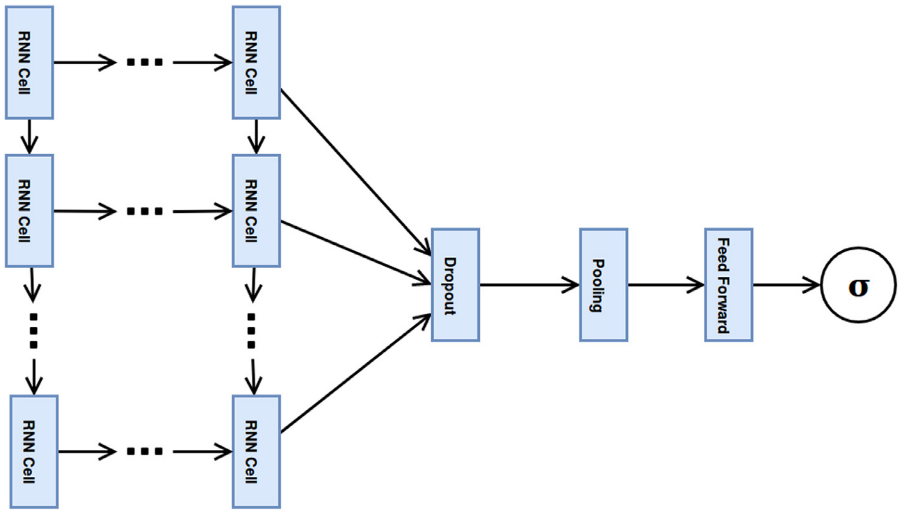

Fig. 2. Proposed RNN architecture.

order to build a RNN classifier that handles the varying lengths of the sessions, it is necessary to zero-pad all sessions to the length of the largest one and pass the sequence lengths to the model such that only relevant fields in the tensor are learned. This process is called *masking*. Masking informs the model that some parts of the input data are actually zero-padded and should be ignored during training.

A single session with a very large number of page views will dramatically increase the size of the input tensor. In prior experiments, we found that using only the first 100 page views does not lead to any loss in predictive performance compared to padding sequences to the actual longest session, which has a size of 396. On the other hand, training with a cutoff of 100 significantly decreases memory consumption.On average, a session in our data set includes about 16 page views. Therefore, setting the cutoff to 100 involves heavy use of zero-padding and might still consume more memory than needed. However, the resources available for this study facilitated the 100 page view cutoff. Furthermore, masking prevents an adverse impact on RNN training from zero-padding. On the other hand, lowering the cutoff further would involve more pruning and thus loss of observed session data. Considering the respective consequences of smaller and larger cutoffs, pretest results, and resource availability, we are confident that the imposed cutoff does not impact the results of the study much.

The data set obtained after preprocessing is split into training and test set. Given the total time horizon of two months, we use sessions from the first month for training and the sessions from the second month for testing. Summary statistics of the preprocessed data are provided in Table 2.

5.2. Feature engineering

To capture a user's behavior during a session, we create several features based on the raw clickstream data. Such features include counts, values and averages of items viewed or put into the basket,

Table 2
Summary statistics of the cleaned and preprocessed data set.

	Training	Test	Total
Sessions	277069	279594	556663
Page views	4333576	4280309	8613885
Fraction of new visitors	0.4255	0.3958	0.4106
Conversion rate	0.0801	0.0835	0.0818

as well as the number of page views or the total session duration until the current page view. We also incorporate dummy features that describe the content of the current. For example, we create a dummy that indicates whether a user currently views a page with service information (e.g., terms of delivery, privacy policy, etc.). Another multinomial feature captures whether the current page contains product information and to which category the focal product belongs.

An advantage of sequence classification is the ability to include features that can only be measured after a page view has been completed, with the time spent on a specific page being an example. We include such a feature when using GRU and LSTM classifiers. However, this feature is not feasible when using clipping at every click. Therefore, benchmark classifiers use an alternative feature capturing the average time spent on previous pages. Apart from measuring behavior within the current user session, we augment sessions with features that capture user behavior in previous shop visits. Note that such information is only available for approximately 59% of all sessions in the data set as some visitors are new or not re-identified. In the latter case, we set feature values to suitable defaults. For example, considering a feature that captures the number of orders across past user visits and a feature measuring the number of days since the user's last visit, we use default values of zero and the maximum value in the training set, respectively. In addition, the data set also includes general information about the session such as the starting hour of the session, the referring page type (e.g., price comparison site, search engine, etc.) or an indicator whether the user accesses the website using a mobile phone.

A nondisclosure agreement with the data donor prohibits revealing more details about the feature set. We suggest that the previous example equips readers with a solid understanding of the type of features. We also refer interested readers to Baumann, Haupt, Gebert, and Lessmann (2019) who provide a comprehensive list of clickstream based features in their online Appendix. As illustrated by the previous examples, the features considered in this paper follow a similar logic. After feature engineering, the final data set contains 106 features. Since most classifiers benefit from scaled inputs, we standardize all numerical features based on their means and standard deviations in the training set.

5.3. Benchmark models

To benchmark the RNN-based sequence classifiers, we consider four well-established SML algorithms as benchmarks. The benchmark classifiers perform clipping at every click. Logistic regression (LR) serves as a first, linear benchmark. Due to its simplicity, its few hyper-parameters and computational efficiency, LR has become the de-facto standard in the industry. Related research also relies on LR (Olbrich & Holsing, 2011; Van den Poel & Buckinx, 2005).

We also consider more complex tree-based ensembles in the form of random forest (RF) and gradient boosting machines (GBM). Their choice is motivated by high predictive performance in various prediction tasks (see, e.g., Lessmann et al. (2019) or Taieb and Hyndman (2014)). Finally, we consider a multi-layer perceptron (MLP), as it is a natural starting point before applying more complex neural network architectures. For this reason, MLP is commonly used to benchmark RNN-based sequence classifiers (Graves, 2012). In the interest of brevity, we omit a detailed overview of the benchmark at this point and refer readers to Hastie, Tibshirani, and Friedman (2009).

5.4. Model selection

We tune classification models using random search and 10-fold cross-validation (Greff, Srivastava, Koutnk, Steunebrink, & Schmidhuber, 2017). When employing cross-validation for the benchmark models, we make sure that all page views belonging to a user session appear in the same fold and keep their original order. This is done to avoid situations where earlier page views of a user session end up in the holdout fold while later page views are used for training. The hyper-parameter combinations considered for each model are given in Table 3.

We randomly evaluate 20 combinations out of all hyper-parameter combinations. This means that for each algorithm 200 different classifiers are fitted. Each of these fits are evaluated with the area under the receiver operating characteristic curve (AUC) on the current holdout fold. The hyper-parameter combination of the model with highest average AUC across the 10 folds is then chosen for subsequent comparisions on the test set.

We apply early stopping for GBM to find the optimal number of trees and for all neural network algorithms to find the optimal number of epochs. For that matter, we keep another 5% of the training data for each fit as a validation set. If the AUC is not increased by an additional five trees or epochs respectively, the model training is stopped. The motivation for early stopping in both cases is twofold. Firstly, it decreases training times of the corresponding algorithms significantly. Secondly, it has been argued that early stopping has a regularization effect and thus, makes the models less prone to overfitting (see Zhao, Yao, and Zhang (2016) for a discussion of the effects on GBM and Goodfellow et al. (2016) on neural networks).

Apart from early stopping, we choose a few other common settings for all neural network models (MLP, GRU, and LSTM). These include the optimizer and the loss function, which we set to Adam (Kingma & Ba, 2014) and binary cross-entropy, respectively.

5.5. Performance measures

In order to assess sequence classifiers, we evaluate the session-level predictions of different models. In the case of the benchmark models, which use clipping at every click, we treat the prediction of the last page view within a session as the global session prediction. This is motivated by the fact that some features incrementally add information to the model. An example is a feature that counts the total value of items put into the shopping basket. Thus, the prediction of the last page view should include more information than the previous ones and therefore be more accurate. Given the session-level predictions (i.e., model estimated conversion probabilities), we plot the false positive rate against the true positive rate, where non-conversion sessions are labeled as positive examples and converted sessions as negative ones. The receiver operating characteristic (ROC) curve is a common instrument to evaluate binary classification models (Fawcett, 2006). We also report the AUC statistic of alternative models, which summarizes the ROC curve in a single number. Our motivation to use ROC Analysis as opposed to other popular approaches such as the f1-score (Toth et al., 2017) is that it allows us to compare probabilistic forecasts of different models without setting a classification threshold.

Table 3
Summary of hyper-parameter tuning for conversion classification. Naming conventions of the hyper-parameter either follow the documentation of the corresponding software package (e.g. *C* for the inversed regularization term) or literature standards (e.g. *mtry* for the number of variables in the random subset at each node (Liaw, Wiener et al., 2002)). An exception is the learning rate, which we abbreviate with *lr*. For detailed information on default settings and the exact meaning of the listed hyper-parameters, we refer the reader to the documentation of the corresponding software packages, namely http://scikit-learn.org *scikit-learn* (Pedregosa et al., 2011), https://tech.yandex.com/catboost/doc/dg/concepts/about-docpage/ *catboost* (Prokhorenkova, Gusev, Vorobev, Dorogush, & Gulin, 2017) and https://pytorch.org/docs/stable/index.html *pytorch*. A comprehensive overview and empirical analysis of these hyper-parameters is also available in a recent paper by Greff et al. (2017).

Model	Implementation	Parameter grid	Best Parameters
LR	scikit-learn	$C = logspace(-3, 3, 30)$	$C = 483.29$
RF	scikit-learn	$mtry = [20, 40, 60]$	$mtry = 40$
		$ntree = [100, 200, 300]$	$ntree = 200$
		$min\_sample\_leaf = [0.0001, 0.001, 0.01]$	$min\_sample\_leaf = 0.001$
GBM	catboost	$max\_depth = [6, 8, 10]$	$max\_depth = 6$
		$lr = [0.1, 0.01, 0.05, 0.005, 0.001]$	$lr = 0.1$
		$boosting\_sample = [0.4, 0.6, 0.8]$	$boosting\_sample = 0.4$
			$n\_tree = 245$
MLP	pytorch	$lr = [0.005, 0.001, 0.0001, 0.0005]$	$lr = 0.0005$
		$dropout\_rate = [0.2, 0.4, 0.6]$	$dropout\_rate = 0.2$
		$hidden\_nodes = [128, 256, 512]$	$hidden\_nodes = 512$
GRU	pytorch	$lr = [0.005, 0.001, 0.0001, 0.0005]$	$lr = 0.0005$
		$dropout\_rate = [0, 0.25, 0.5]$	$dropout\_rate = 0.2$
		$rnn\_layer = [1, 2, 3]$	$rnn\_layer = 2$
		pooling=[None, AvgPooling, MaxPooling]	*pooling*=None
LSTM	pytorch	$lr = [0.005, 0.001, 0.0001, 0.0005]$	$lr = 0.005$
		$dropout\_rate = [0, 0.25, 0.5]$	$dropout\_rate = 0.6$
		$rnn\_layer = [1, 2, 3]$	$rnn\_layer = 1$
		pooling=[None, AvgPooling, MaxPooling]	*pooling*=None

Table 4
Revenue gain matrix of an e-coupon targeting model. Columns depict the actual class label of user sessions. Rows depict class predictions (and implied actions). Benefits and costs arise only if a coupon is issued.

		Actual label	
		Buyer	Non-Buyer
Prediction	Buyer (no coupon)	0	0
	Non-Buyer (issue coupon)	$-(r_i \times \delta)$	$p \times r_i \times (1-\delta)$

5.5.1. Revenue-based evaluation

In addition to ROC-Analysis, we evaluate classification models in terms of their value for marketing campaign planning. To that end, we consider the specific context of an e-coupon system and evaluate the costs and benefits resulting from targeting customers according to model-based predictions. In the context of our study, a targeting model predicts whether shop visitors are going to purchase items in their current session. Classifying a visitor as a buyer implies that we expect this visitor to buy anyway and thus translates into the decision to not issue a coupon. Similarly, we expect visitors that the model classifies as non-buyers to not purchase in their session. Issuing a coupon to these visitors might alter their expected behavior, make them buy, and increase shop revenue. Using the data from the test set, we have access to the true labels of shop visitors, that is whether they bought in their session. Based on the predicted and actual class labels, we propose the revenue gain matrix shown in Table 4, where we assume that benefits and costs arise only if a coupon is issued. This implies that we consider a base scenario in which no targeting model is deployed and no coupons are used to stimulate additional sales.

Table 4 considers the following scenario. Let r_i be the revenue from selling items to customer i. An issued coupon reduces r_i by a fraction δ of the original price of all purchased items. In our analysis we will set $\delta = 0.1$, i.e. a customer receiving the coupon only needs to pay 90% of the total price of her purchased items. In addition, let p denote the probability that a customer who did not plan to purchase is turned into a buyer by the price reduction of the coupon. We will refer to p as redemption probability from hereon. If on the other hand, customers are targeted although they have already planed a purchase, it is assumed that the coupon is used without changing the initial purchase plan. In the other two possible cases, there will be no direct change in revenue, but note that falsely predicting purchase sessions carries opportunity costs of not targeting actual non-purchase customers. Given this scenario, the gain in revenue achieved by an e-coupon campaign can be calculated as:

$$\Delta\pi = \sum_{i=1}^{N_{tp}} p \times r_i \times (1-\delta) - \sum_{j=1}^{N_{fp}} r_j \times \delta, \tag{15}$$

where N_{tp} is the number of true positive and N_{fp} the number of false positive sessions classified.

5.5.2. Estimating Potential Order Values

In order to use Eq. (15) for model evaluation in practice requires specifying two quantities: the redemption probability p and the potential order values r_i of all N_{tp} targeted non-buyers. The latter can be estimated by a regression model using the data at hand. In particular, we fit a regression model on sessions in the training set that ended with a purchase using the observed purchase values as the target variable. To assess the quality of the predicted order values, all unseen purchase sessions in the test set are used as holdout set. An advantage of neural networks is that their architecture can be easily adapted to different targets. We make use of this flexibility and consider the same GRU and LSTM architectures that were previously proposed for the classification problem with the exception that we are leaving out the final sigmoid layer and change the loss function to mean squared error (MSE). In order to check whether the results of the proposed regressors are reasonable, we construct a naive baseline predictor that always predicts the median order value of all purchase sessions in the training set. The benchmark is motivated by Baumann et al. (2019), who use this approach to estimate order values. In addition, we also fit a ridge regression (Ridge), random forest regression (RFR), gradient boosting regression (GBR) and an MLP regressor to the training set to benchmark the RNN regressors. As in the classification task, clipping at every click is used. This time, however, page views are assigned to the final order value of their session. Hyper-parameters of the considered regression models are also tuned in the same fashion as in the classification case, i.e. using random search with 10-fold cross-validation. Table 5 summarizes the corresponding hyperparameters. It is important to note that tuning order value estimation methods using cross-validation avoids leakage of information from the test set.

5.5.3. Constructing revenue gain charts

While it is possible to estimate the potential order values of targeted customers, it is difficult to find a reasonable strategy to estimate the redemption probability from the data at our disposal. This is due the fact that we have no data on previous e-coupon campaigns. Facing such a cold start problem, we calculate gains in

Table 5
Summary of hyper-parameter tuning for order value prediction.

Model	Implementation	Parameter grid	Best Parameters
Ridge	scikit-learn	$\alpha = logspace(-5, 5, 30)$	$\alpha = 3.45 \times 10^{-8}$
RFR	scikit-learn	$mtry = [20, 40, 60]$	$mtry = 20$
		$ntree = [100, 200, 300]$	$ntree = 200$
		$min\_sample\_leaf = [0.0001, 0.001, 0.01]$	$min\_sample\_leaf = 0.0001$
GBR	catboost	$max\_depth = [6, 8, 10]$	$max\_depth = 6$
		$lr = [0.1, 0.01, 0.05, 0.005, 0.001]$	$lr = 0.1$
		$boosting\_sample = [0.4, 0.6, 0.8]$	$boosting\_sample = 0.4$
MLP	pytorch	$lr = [0.005, 0.001, 0.0001, 0.0005]$	$lr = 0.001$
		$dropout\_rate = [0.2, 0.4, 0.6]$	$dropout\_rate = 0.4$
		$hidden\_nodes = [128, 256, 512]$	$hidden\_nodes = 128$
GRU	pytorch	$lr = [0.005, 0.001, 0.0001, 0.0005]$	$lr = 0.0005$
		$dropout\_rate = [0.2, 0.4, 0.6]$	$dropout\_rate = 0.6$
		$rnn\_layer = [1, 2]$	$rnn\_layer = 1$
		$pooling$=[None, AvgPooling, MaxPooling]	$pooling$=MaxPooling
LSTM	pytorch	$lr = [0.005, 0.001, 0.0001, 0.0005]$	$lr = 0.0005$
		$dropout\_rate = [0.2, 0.4, 0.6]$	$dropout\_rate = 0.6$
		$rnn\_layer = [1, 2]$	$rnn\_layer = 1$
		$pooling$=[None, AvgPooling, MaxPooling]	$pooling$=None

campaign revenue for different redemption probabilities. For that matter, we choose an equidistant interval of ten values in the range from 0.01 to 0.1. Finally, before computing an estimate for $\Delta\pi$, the probability outcomes of the classifiers need to be converted into actual decision, i.e. 0 and 1 labels, by setting a probability threshold τ. Assuming that coupons are issued to every non-conversion session, τ can be used to control the fraction of the population that is targeted by the campaign. For instance, in an extreme case where $\tau = 0$, every session will be targeted, while $\tau = 1$ implies that no coupons are issued. Therefore, instead of computing $\Delta\pi$ for a fixed probability threshold, we will use τ to construct revenue gain charts, i.e. plotting the targeted population of a campaign against the resulting revenue gain. Related versions of such charts are widely adopted in the literature to visualize the impact of the classifier's decision on marketing campaigns, e.g. Park and Park (2016) or Sołtys, Jaroszewicz, and Rzepakowski (2015). In a gain char, the diagonal line between the two extreme cases represents the baseline revenue gain achieved by random targeting (Radcliffe, 2007). To enable a comparison of the models on the basis of all ten redemption probability values, we compress each curve into a single number using the *Gini coefficient*. In the context of revenue gain charts, the Gini coefficient is defined as the ratio of two areas. The numerator is the area between the actual revenue gain curve and the diagonal corresponding to random targeting. The denominator is the same area but for the optimal revenue gains curve (Radcliffe & Surry, 2011). Consequently, a perfect model results in a Gini coefficient of 1, while random targeting leads to a score of 0.

6. Results

In the following, we first discuss the results of the order value prediction task. Next, we elaborate on the predictive performance of different clickstream classifiers. We conclude the section with a brief analysis of prediction runtimes. Runtimes are crucial in practice because the calculation of model predictions delays the loading of a web page. Therefore, putting an e-coupon system into production requires models to produce predictions with low latency.

6.1. Predictive performance of order value regressors

Table 6 summarizes the predictive performance of order value predictions using standard regression metrics in the form of root mean squared error (RMSE), mean absolute error (MAE), median absolute deviation (MAD) and R^2. Note that we do not report the results for ridge regression and gradient boosting regression as these models fail to outperform the naive median baseline in any of the reported measures. In fact, only the two RNN-based architectures manage to outperform the naive model on all measures. Out of these two architectures, the GRU seems to perform slightly better than the LSTM. In spite of this, it is difficult to determine which RNN model is more suitable for predicting the potential order values of non-purchase sessions as differences in error measures are marginal. Fig. 3 displays a kernel density plot of predicted order values from both models and from the actual orders of purchase sessions. This plot shows that the values of the true

Table 6
Performance of regression models.

Model	RMSE	MAE	MAD	R^2
Naive Model	102.78	62.14	37.97	0
Random Forest	93.26	61.40	41.35	0.1935
MLP	91.95	59.23	39.88	0.2104
GRU	**73.72**	**45.10**	**27.59**	**0.4337**
LSTM	73.87	45.68	27.77	0.4315

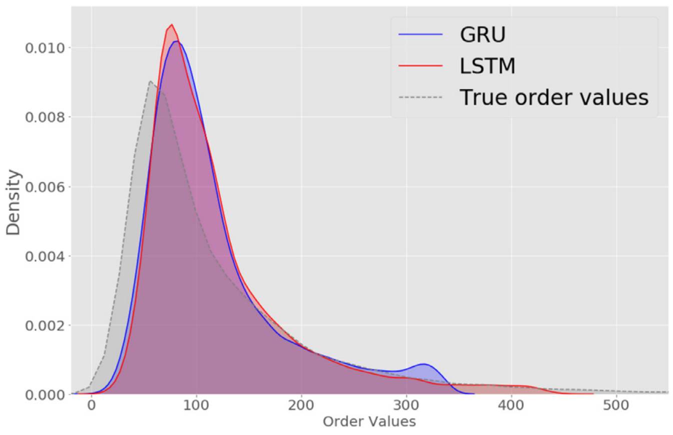

Fig. 3. Actual vs. Predicted OV.

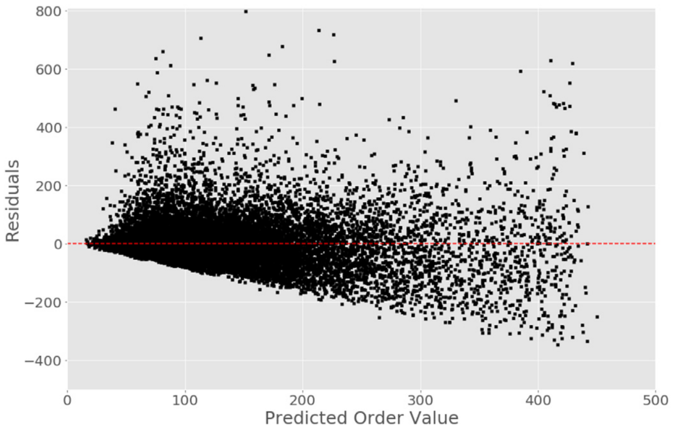

Fig. 4. Residual Plot (LSTM).

purchases have a higher variance and include more extreme cases than the predicted order values of both models. In contrast to this, order value predictions of GRU and LSTM are more concentrated around values between 60 and 100. Another interesting point revealed by the density plot concerns the right tail of the GRU prediction outcomes. The largest order values predicted by the GRU are scattered around a small area slightly below the maximum of 338.43, producing a bump at the right end of the distribution. The predictions by the LSTM on the other hand, have a higher maximum and large values are more spread out. The difference between GRU and LSTM outcomes becomes even more evident when looking at the residual plots of both models. Fig. 5 shows that the chosen GRU architecture leads to an upper limit of prediction values. The LSTM residual plot in Fig. 4 does not show such a limit. We suggest that this difference comes from the fact that the GRU uses a max-pooling layer whereas the LSTM architecture does not. Recall that we leave the decision whether to include a pooling layer as well as the specific form of pooling to the model selection mechanism (see Table 5).

Generally, we observe that the GRU outputs more conservative predictions as the sum of the differences between actual order values and predicted ones is positive. The opposite is true for the LSTM. Assuming that high order values for customers that are turned into buyers by the coupon occur more rarely than for regular buyers, a more pessimistic prediction output might be more desirable in our scenario. As a consequence, the GRU will be used as a final model to predict the potential order values of targeted non-purchase sessions.

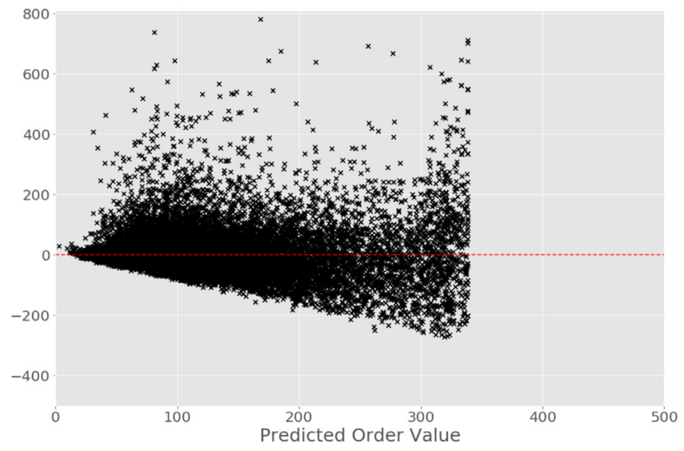

Fig. 5. Residual Plot (GRU).

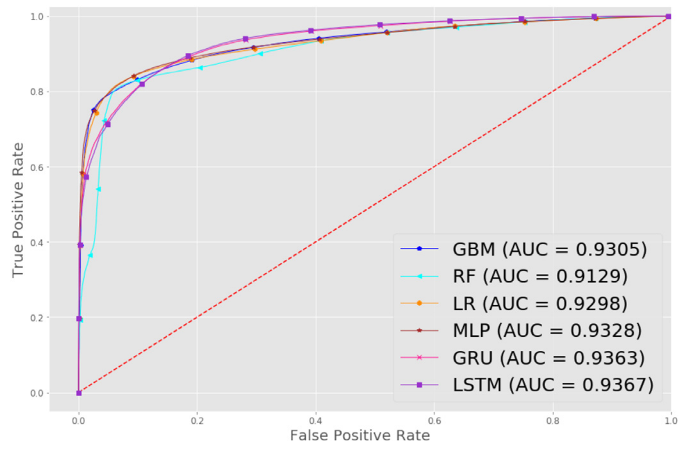

Fig. 6. ROC curve ofconversion classifiers.

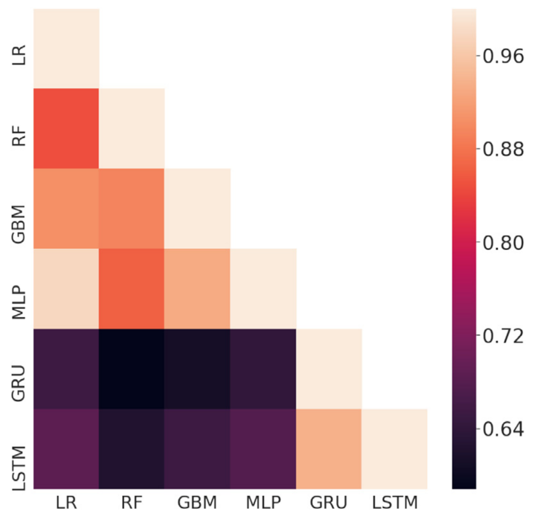

Fig. 7. Correlation of classifier outcomes.

6.2. *Predictive performance of conversion classifiers*

We split the analysis of conversion models' predictive performance into three parts. First, we employ a ROC analysis to obtain general insights into the predictive power of each model. It follows a discussion on the diversity of classifier outcomes. Finally, we adopt the proposed revenue gain estimation methodology to evaluate classifiers in the context of an e-coupon scenario.

6.2.1. *ROC Analysis*

Fig. 6 shows the ROC curves of the considered classifiers. From this plot, we observe that the curves of the two *sequence classifiers*, namely LSTM and GRU, co-move and behave differently than the curves of the four benchmark models that perform *clipping at every click*. In particular, the slopes of the RNN model curves are considerably lower for smaller probability thresholds τ than the slopes of the benchmark model curves. This means that we have to trade a higher increase in the false positive rate to accomplish a higher true positive rate. However, both RNN curves exceed the curves of the benchmark models at some point meaning that in comparison to the benchmark models higher true positive rates are achieved for larger values of τ. In terms of AUC, LSTM and GRU outperform all other models, but differences seem to be marginal. An exception to this is random forest (RF) which performs much worse than all other models on our data.

6.2.2. *Diversity of classifier outcomes*

The different shapes of the ROC curves suggest that the two clickstream modeling approaches capture different patterns. In line with this, we find that the output of the two RNN classifiers are highly correlated, while the correlation with other classifier outputs is moderate. Fig. 7 visualizes the values of the pair-wise Pearson correlation coefficients between different classifier outcomes in a heatmap. Lighter fields indicate a higher correlation between two classifier outputs. We observe the lowest correlation between the tree-based ensembles and the GRU (0.58 with RF and 0.61 with GBM). Furthermore, ROC analysis suggested that GBM and GRU are among the best performing models. Hence, we have a situation in which the outcomes of two strong models are relatively dissimilar. In such situations, it might be beneficial to combine both models by building an ensemble classifier (Abelln & Castellano, 2017). In order to explore this potential benefit, we consider a simple approach that averages the two prediction outcomes, i.e. prediction outcome of i-th session is computed with

$$\hat{y}_i^{ens} = \frac{\left(\hat{y}_i^{gru} + \hat{y}_i^{gbm}\right)}{2}, \tag{16}$$

where $\hat{y}_i^{gru}$ and $\hat{y}_i^{gbm}$ are the probability outcomes of the GRU and GBM for session i, respectively. The ROC curve of such an ensemble classifier is displayed in Fig. 5. Given this curve, we conclude that the simple ensemble strategy is effective and outperforms the best individual models with much higher AUC. Note that this also holds for other combinations of RNN and benchmark classifiers, but the illustrated combination of GRU and GBM performs best. As averaging the classifier outcomes seems to be sufficient, we omit applying more systematic ensemble selection strategies such as ensemble selection (Lustosa Filho, Canuto, & Santiago, 2018) or stacking (Xia, Liu, Da, & Xie, 2018).

6.2.3. *Revenue gain analysis*

Employing our strategy to estimate revenue gains, we find that the resulting curves behave similarly to the ROC curves in the previous analysis. In particular, the revenue gains of all benchmark models progress similarly as the population is varied. Therefore, Figs. 8–14 only show the curves of the best benchmark model, i.e. MLP, and the curves of GRU, LSTM and the ensemble presented above. Moreover, we omit to display revenue gain charts for higher redemption probabilities p than 6%, because these charts follow the same trends shown in the provided figures.

Generally, the revenue gain charts reveal that each model performs much better than random targeting for any fraction of the whole customer population. A comparison of the curves suggests that the benchmark models are slightly preferable for smaller populations while the RNNs tend to achieve higher campaign revenues

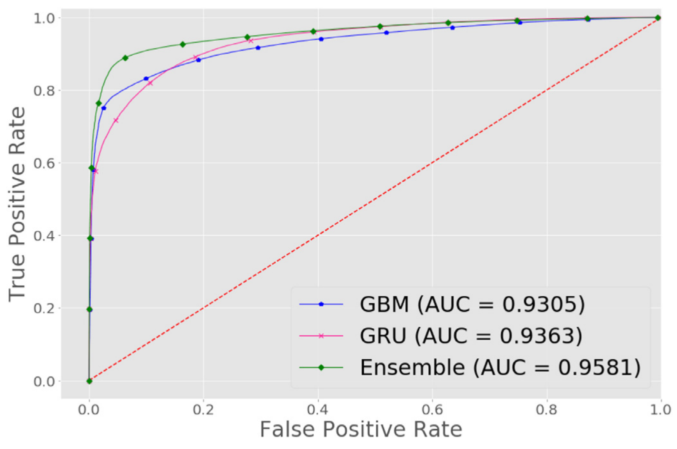

Fig. 8. ROC curve with ensemble.

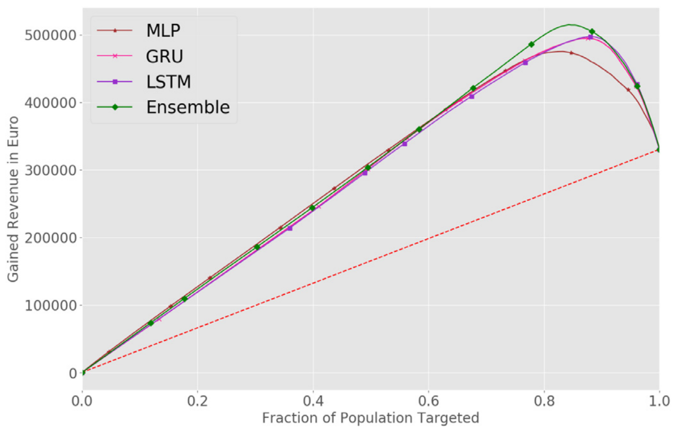

Fig. 11. $p = 0.03$ and $\delta = 0.1$.

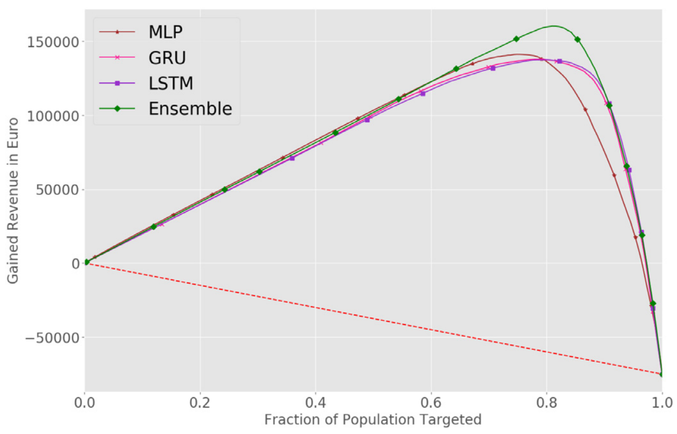

Fig. 9. $p = 0.01$ and $\delta = 0.1$.

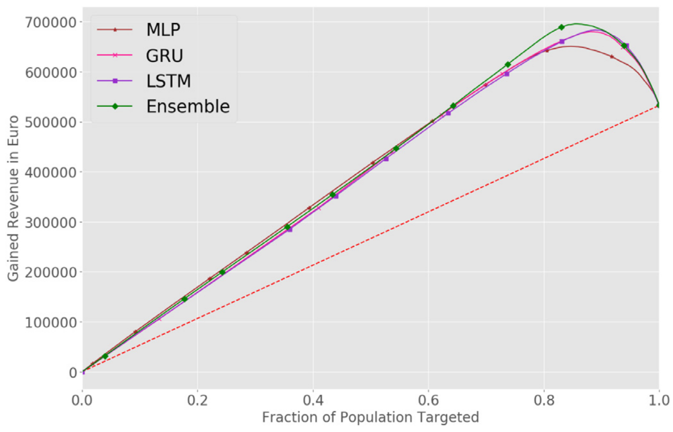

Fig. 12. $p = 0.04$ and $\delta = 0.1$.

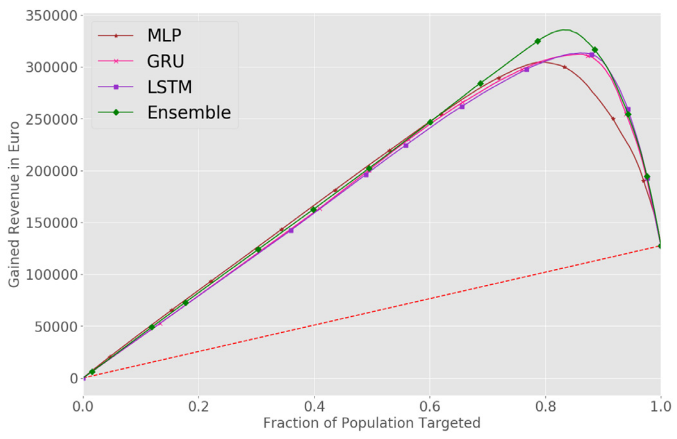

Fig. 10. $p = 0.02$ and $\delta = 0.1$.

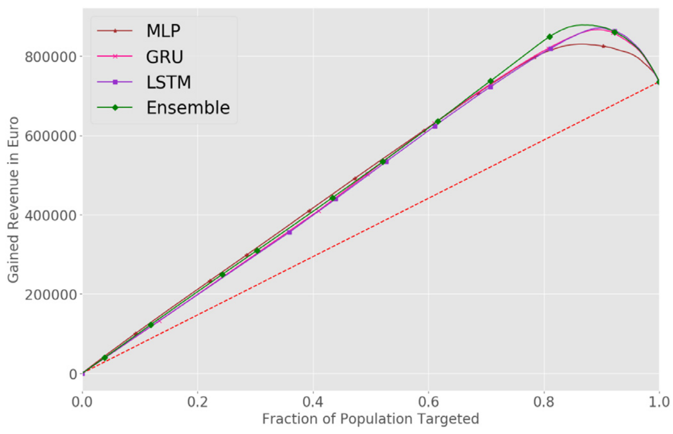

Fig. 13. $p = 0.05$ and $\delta = 0.1$.

when the majority of the population is targeted. The latter observation only holds if p is at least 0.02. As expected, the ensemble classifier balances these two characteristics by accomplishing a comparable revenue gain as the MLP for smaller populations and having the highest maximum revenue gain. This observation is true for all charts constructed. In line with this, the ensemble also has the largest Gini coefficients for every p reported in Table 7. Based on this metric the two RNN classifiers seem to outperform the benchmark models. Nevertheless, this does not imply that LSTM and GRU are preferable for the focused e-coupon application. Drawing such a conclusion neglects the fact that different parts of the gain chart carry different relevance for marketing. In offline marketing, campaigns are typically limited to target a small fraction of customers, which comes from the fact that the costs of contacting customers via offline channels are sizeable (De Bock & Van den Poel, 2011). Although contact costs are negligible in digital marketing, a preference for smaller campaign sizes seems still plausible in our setting because issuing an excessive amount of coupons might dilute the brand image of the online shop and decrease customers' willingness-to-pay in the long run. Taking this point into consideration, we conclude that RNNs and the ensemble classifier are not superior in general for the focal

Table 7
Gini Coefficients for different redemption probabilities.

	Gini coefficient (area under the revenue gain curve)									
p	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
LR	0.3665	0.5550	0.6193	0.6506	0.6694	0.6819	0.6908	0.6975	0.7028	0.7069
RF	0.2623	0.4452	0.5729	0.6266	0.6447	0.6563	0.6646	0.6708	0.6756	0.6795
GBM	0.3400	0.5404	0.5983	0.6273	0.6749	0.7005	0.7187	0.7323	0.7430	0.7515
MLP	0.3689	0.5567	0.6267	0.6625	0.6840	0.6984	0.7087	0.7163	0.7223	0.7270
GRU	0.4632	0.6633	0.7300	0.7633	0.7833	0.8067	0.8062	0.8133	0.8189	0.8234
LSTM	0.4824	0.6808	0.7469	0.7800	0.7999	0.8131	0.8226	0.8296	0.8351	0.8396
Ensemble	**0.5052**	**0.7022**	**0.7545**	**0.7956**	**0.8143**	**0.8268**	**0.8357**	**0.8424**	**0.8476**	**0.8517**

application as they led to less revenue for smaller target populations in some experimental settings. At this point, it is important to note that we arrive at this conclusion only because of examining the monetary implications of the learning models using the proposed profit-based evaluation framework. Considering only the results concerned with predictive accuracy, which we measure in terms of the AUC, the conclusion would have been that RNN-based approaches are strictly superior for clickstream classification and e-coupon targeting, respectively. The two different pictures that we observe from the results in terms of AUC and campaign profit highlight the merit of the profit-based evaluation.

6.3. *Prediction runtimes*

In addition to predictive performance, we also examine the runtime that is necessary to compute classifier predictions. Prediction times are crucial in real-time applications. Even a model with perfect accuracy is effectively useless for an e-coupon system, or any other live scenario for that matter, if it fails to produce predictions sufficiently fast. Fig. 15 illustrates the distribution of seconds spent on a page by the users in the data set. Roughly 10% of all page views take 3 seconds or less. Given that additional computations such as attribute measuring and feature calculation is necessary before feeding the data to the learned decision function and that the user needs to be able to anticipate the existence of the displayed coupon, a suitable model should be able to output prediction within a few milliseconds (ms) in order to be effective. Using the trained models, we measure their runtimes for predicting sessions in the test set. To ensure a fair comparison, all algorithms are executed on the same hardware, i.e. a machine running on Linux Ubuntu 16.04 with Intel Core i7-6700K CPU and 32GB RAM. Fig. 16 summarizes the measured prediction times for each classifier in a boxplot. Note that RF is excluded from this plot as its prediction runtime average with approximately 11ms is more than five times higher than the average of the second slowest model. Unlike RF, GBM predictions are competitively fast even though each data point needs to be processed through 245 decision trees. The reason for the large difference between the two tree-ensembles is that trees in GBM are restricted to a maximum depth of 6 and thus, are much more shallow. Despite the fact that RNNs are slower than LR, GBM, and MLP, their runtimes are not prohibitively slow. The LSTM architecture has an average of 1.38ms and therefore is a lit-

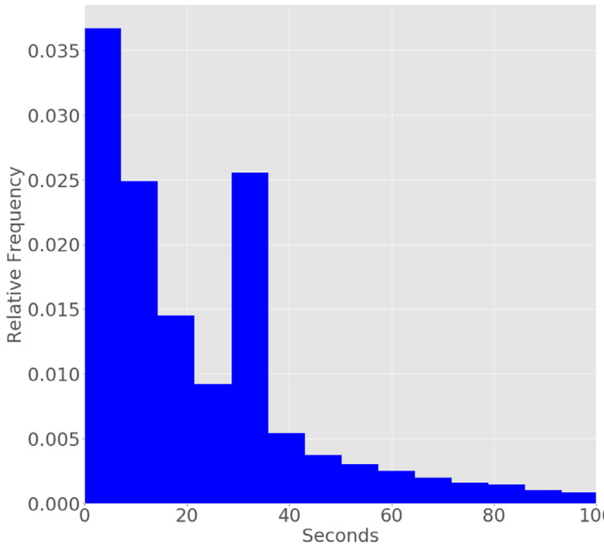

Fig. 15. Histogram seconds on page.

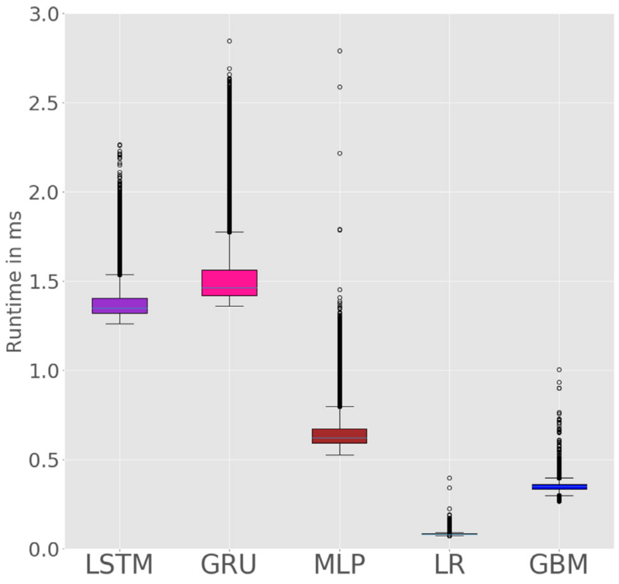

Fig. 16. Prediction runtime (in ms).

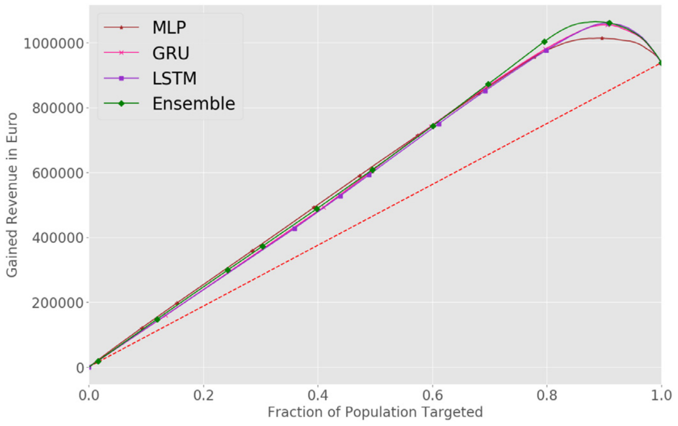

Fig. 14. $p = 0.06$ and $\delta = 0.1$.

tle faster than the GRU which takes 1.52ms on average to predict a session. Furthermore, the boxplot reveals that the runtimes of the GRU have higher variance. This is not surprising. The GRU architecture includes two recurrent layers and is more complex than the architecture of the LSTM, which has only one recurrent layer. Based on the runtimes of GRU and GBM, we can also conclude that the ensemble of both classifiers is sufficiently fast for a live scenario. An average runtime of 1.87ms of the ensemble is still acceptable for an e-couponing system.

7. Conclusion

We identified previous approaches to derive user behavior predictions from clickstream data to suffer conceptual shortcomings. Deep learning methods for sequence classification promise an appropriate modeling of the sequential structure of clickstream data. We apply corresponding methods to solve a conversion classification problem. This task is relevant for e-commerce and digital marketing and served the paper as a testbed to assess different variants of RNNs. Empirical results revealed that RNN-based sequence classifiers predict user conversions more accurately than previous modeling approaches. To check whether superiority accuracy translates into monetary value, we developed a framework to measure the profitability of a conversion classifier when used for targeting digital coupons. Applying our profit-based evaluation approach, we found that sequence classifiers perform competitive but not better than previous models. We also observed the two types of models to capture different patterns in data and exhibit little forecast correlation. Based on this result, we developed an ensemble of sequence- and conventional classifiers and found it to provide the most accurate and profitable prediction across all models considered in the study.

7.1. Implications

The empirical results have several implications for industry and academia. From a managerial perspective, new (deep learning) methodologies to process sequential clickstream data are interesting because this data source is routinely used in web usage mining and real-time targeting. RNNs are computationally more demanding than the conventional supervised learners used in the industry. Deploying new models is also associated with organizational challenges concerning, for example, learning new analytical methods. In this regard, practitioners might find comfort in our results that industry workhorses like logistic regression or a more powerful random forest classifier are not strictly inferior to deep learning-based methods; despite the latter being more suitable from a conceptual point of view. In view of our results, continuing to use the clipping at every click approach together with a supervised learner is a viable strategy. However, in settings where predictive accuracy is crucial and the monetary value of accurate conversion predictions exceeds the cost of learning and deploying new modeling techniques with a large margin, our results suggest a different course of action. Given the low forecast correlation between sequence and conventional classifiers, deploying a corresponding ensemble model can be expected to raise accuracy and add to the bottom line, at least for the coupon targeting setting which we consider.

A possibly important implication of our study for academia concerns the distinction between predictive accuracy, which we measure in terms of the AUC, and business value, which we approximate using the proposed profit-based evaluation framework. Just looking at the AUC, it would have been easy to judge RNN-based approaches superior to conventional supervised learners. The profit-based evaluation assessed models from a different angle. This provided additional insight and motivated developing the ensemble classifier, which proved to be the most powerful approach overall. Against this background, we encourage future studies to also employ conceptually different performance measures, which, in combination, can give a more comprehensive picture of the adequacy of alternative prediction models.

7.2. Limitations and future research

The paper exhibits some limitations that open up opportunities for future research. Starting with the employed data, we miss certain pieces of information, which required making assumptions are simplifications. Consider the profit-oriented evaluation approach, which we propose in Section 5.5. It requires an estimate of the probability of coupon redemption. Information on redemption probabilities was not available in our data set. Following Baumann et al. (2019), we addressed this issue by examining model performance across a set of plausible redemption probabilities. This approach assumes redemption probabilities to be constant across user sessions. Future research could contribute additional empirical evidence from field experiments, which capture actual redemption probabilities and heterogeneity in redemption behavior across users. Corresponding results, for example from A/B testing, would provide a more realistic picture of the monetary implications of coupon campaigns and how these differ with the employed targeting models.

Designing an empirical study requires making certain decisions and these may impact the observed results. For example, the length of the clickstream varies across user sessions. Approaches to standardize session length in the form of zero-padding and pruning facilitate the use of RNNs for sequence classification. We secured the viability of corresponding choices in pre-tests but cannot rule out that results would vary when imposing different thresholds on the maximum length of a session. This highlights the value of replication studies, which could also revisit certain design decisions, and are a fruitful avenue for future work.

We also acknowledge a more general limitation of our paper, which stems from the choice of conversion classification as a vehicle for e-coupon targeting. Regardless of the clickstream modeling strategy, conversion classifiers can only support targeting decisions. They offer no recommendation on how large a discount should be (i.e., coupon value). Future research could seek richer ways to support marketing decision-making by identifying the optimal marketing action for each user.

In terms of the employed methodology, deep learning is an active field of research and new modeling approaches, including approaches for sequence classification, keep appearing. Future research can thus revisit the RNN frameworks we consider and compare them to alternative modeling regimes. For example, network architectures with different types of layers could facilitate learning general session information from static features (e.g., user device) while learning navigational features from dynamic features (e.g., number of pages visited in the session). Moreover, given the success of CNNs in other applications (Kraus, Feuerriegel, & Oztekin, 2020), it might be worth considering a convolutional layer to learn a better feature representation before processing the sessions through recurrent layers. Combinations of CNN and LSTMs have been considered in the scope of computer vision (Bai, Tang, & An, 2019) and financial time series modeling (Hoseinzade & Haratizadeh, 2019) and could prove useful for clickstream data modeling.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswa.2020.113342.

Credit authorship contribution statement

Dennis Koehn: Conceptualization, Methodology, Software, Writing - original draft. **Stefan Lessmann:** Conceptualization, Validation, Writing - review & editing, Supervision. **Markus Schaal:** Conceptualization, Validation, Project administration.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … Isard, M., et al. (2016). Tensorflow: a system for large-scale machine learning. In *OSDI: 16* (pp. 265–283).

Abelln, J., & Castellano, J. G. (2017). A comparative study on base classifiers in ensemble methods for credit scoring. *Expert Systems with Applications, 73*, 1–10. doi:10.1016/j.eswa.2016.12.020.

Arora, S., & Warrier, D. (2016). Decoding fashion contexts using word embeddings. KDD workshop on machine learning meets fashion.

Bai, S., Tang, H., & An, S. (2019). Coordinate cnns and lstms to categorize scene images with multi-views and multi-levels of abstraction. *Expert Systems with Applications, 120*, 298–309.

Barkan, O., Brumer, Y., & Koenigstein, N. (2016). Modelling session activity with neural embedding. *Recsys posters*.

Baumann, A., Haupt, J., Gebert, F., & Lessmann, S. (2018). Changing perspectives: Using graph metrics to predict purchase probabilities. *Expert Systems with Applications, 94*, 137–148.

Baumann, A., Haupt, J., Gebert, F., & Lessmann, S. (2019). The price of privacy. *Business & Information Systems Engineering, 61*, 413–431.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks, 5*(2), 157–166.

Bertsimas, D. J., Mersereau, A. J., & Patel, N. R. (2003). Dynamic classification of online customers. In *Proceedings of the 2003 SIAM international conference on data mining* (pp. 107–118). SIAM.

Bogina, V., & Kuflik, T. (2017). Incorporating dwell time in session-based recommendations with recurrent neural networks. In CEUR workshop proceedings*: 1922* (pp. 57–59).

Borges, J., & Levene, M. (2007). Evaluating variable-length markov chain models for analysis of user web navigation sessions. *IEEE Transactions on Knowledge and Data Engineering, 19*(4), 441–452.

Bucklin, R. E., & Sismeiro, C. (2009). Click here for internet insight: Advances in clickstream data analysis in marketing. *Journal of Interactive Marketing, 23*(1), 35–48.

Chan, T., Joseph, I., Macasaet, C., Kang, D., Hardy, R. M., Ruiz, C., … Hannon, P., et al. (2014). Predictive models for determining if and when to display online lead forms. In AAAI (pp. 2882–2889).

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint, arXiv:1409.1259.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint, arXiv:1412.3555.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2015). Gated feedback recurrent neural networks. In *International conference on machine learning* (pp. 2067–2075).

De Bock, K. W., & Van den Poel, D. (2011). An empirical evaluation of rotation-based ensemble classifiers for customer churn prediction. *Expert Systems with Applications, 38*(10), 12293–12301. doi:10.1016/j.eswa.2011.04.007.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters, 27*(8), 861–874.

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM. Ninth international conference on artificial neural networks (ICANN 99). IET.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning*. MIT Press, Cambridge.

Graves, A. (2012). Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks* (pp. 5–13). Springer.

Greff, K., Srivastava, R. K., Koutnk, J., Steunebrink, B. R., & Schmidhuber, J. (2017). Lstm: A search space odyssey. *IEEE Transaction on Neural Networks and Learning Systems, 28*(10), 2222–2232.

Gubela, R. M., Lessmann, S., Haupt, J., Baumann, A., Radmer, T., & Gebert, F. (2017). Revenue uplift modeling. In Proceedings of the 38th international conference on information systems (ICIS). AIS.

Gubela, R. M., Lessmann, S., & Jaroszewicz, S. (2019). Response transformation and profit decomposition for revenue uplift modeling. *European Journal of Operational Research, 283*(2), 647–661.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). The elements of statistical learning. *Springer Series in Statistics* (2nd). New York: Springer.

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based recommendations with recurrent neural networks. arXiv preprint, arXiv:1511.06939.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Hoseinzade, E., & Haratizadeh, S. (2019). Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications, 129*, 273–285. doi:10.1016/j.eswa.2019.03.029.

Iwanaga, J., Nishimura, N., Sukegawa, N., & Takano, Y. (2016). Estimating product–choice probabilities from recency and frequency of page views. *Knowledge-Based Systems, 99*, 157–167.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint, arXiv:1412.6980.

Kraus, M., Feuerriegel, S., & Oztekin, A. (2020). Deep learning in business analytics and operations research: Models, applications and managerial implications. *European Journal of Operational Research, 281*, 628–641.

Lakshminarayan, C., Kosuru, R., & Hsu, M. (2016). Modeling complex clickstream data by stochastic models: Theory and methods. In *Proceedings of the 25th international conference companion on world wide web* (pp. 879–884). International World Wide Web Conferences Steering Committee.

Lee, J. Y., & Dernoncourt, F. (2016). Sequential short-text classification with recurrent and convolutional neural networks. arXiv preprint, arXiv:1603.03827.

Lessmann, S., Haupt, J., Coussement, K., & De Bock, K. W. (2019). Targeting customers for profit: An ensemble learning framework to support marketing decision-making. *Information Sciences*. online first. doi: 10.1016/j.ins.2019.05.027.

Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R News, 2*(3), 18–22.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. arXiv preprint, arXiv:1506.00019.

Lustosa Filho, J. A. S., Canuto, A. M. P., & Santiago, R. H. N. (2018). Investigating the impact of selection criteria in dynamic ensemble selection methods. *Expert Systems with Applications, 106*, 141–153.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint, arXiv:1301.3781.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., & Khudanpur, S. (2010). Recurrent neural network based language model. *Eleventh annual conference of the international speech communication association*.

Moe, W. W. (2003). Buying, searching, or browsing: Differentiating between online shoppers using in-store navigational clickstream. *Journal of Consumer Psychology, 13*(1-2), 29–39.

Moe, W. W., & Fader, P. S. (2004). Dynamic conversion behavior at e-commerce sites. *Management Science, 50*(3), 326–335.

Montgomery, A. L., Li, S., Srinivasan, K., & Liechty, J. C. (2004). Modeling online browsing and path analysis using clickstream data. *Marketing Science, 23*(4), 579–595.

Olbrich, R., & Holsing, C. (2011). Modeling consumer purchasing behavior in social shopping communities with clickstream data. *International Journal of Electronic Commerce, 16*(2), 15–40.

Park, C. H., & Park, Y.-H. (2016). Investigating purchase conversion by uncovering online visit patterns. *Marketing Science, 35*(6), 894–914.

Park, J., & Chung, H. (2009). Consumers travel website transferring behaviour: analysis using clickstream data-time, frequency, and spending. *The Service Industries Journal, 29*(10), 1451–1463.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., … Lerer, A. (2017). Automatic differentiation in PyTorch. *Nips 2017 workshop autodiff decision program chairs*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Van den Poel, D., & Buckinx, W. (2005). Predicting online-purchasing behaviour. *European Journal of Operational Research, 166*(2), 557–575.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2017). Catboost: unbiased boosting with categorical features. arXiv preprint, arXiv:1706.09516.

Radcliffe, N. (2007). Using control groups to target on predicted lift: Building and assessing uplift models. *Direct Marketing Journal, 1*, 14–21.

Radcliffe, N. J., & Surry, P. D. (2011). Real-world uplift modelling with significance-based uplift trees. *Stochastic Solutions*. White Paper TR-2011-1.

Reimers, I., & Xie, C. (2019). Do coupons expand or cannibalize revenue? evidence from an e-market. *Management Science, 65*(1), 286–300.

Rojas, R. (1996). *Neural networks: A systematic introduction*. Springer Science & Business Media.

Shapoval, K., & Setzer, T. (2018). Next-purchase prediction using projections of discounted purchasing sequences. *Business & Information Systems Engineering, 60*(2), 151–166.

Sismeiro, C., & Bucklin, R. E. (2004). Modeling purchase behavior at an e-commerce web site: A task-completion approach. *Journal of Marketing Research, 41*(3), 306–323.

Sołtys, M., Jaroszewicz, S., & Rzepakowski, P. (2015). Ensemble methods for uplift modeling. *Data Mining and Knowledge Discovery, 29*(6), 1531–1559.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.

Suh, E., Lim, S., Hwang, H., & Kim, S. (2004). A prediction model for the purchase probability of anonymous customers to support real time web marketing: a case study. *Expert Systems with Applications, 27*(2), 245–255.

Taieb, S. B., & Hyndman, R. J. (2014). A gradient boosting approach to the kaggle load forecasting competition. *International Journal of Forecasting, 30*(2), 382–394.

Tamhane, A., Arora, S., & Warrier, D. (2017). Modeling contextual changes in user behaviour in fashion e-commerce. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 539–550). Springer.

Toth, A., Tan, L., Di Fabbrizio, G., & Datta, A. (2017). Predicting shopping behavior with mixture of RNNs. In *Proceedings of the 2017 SIGIR workshop on e-commerce (eCom)*.

VanderMeer, D., Dutta, K., Datta, A., Ramamritham, K., & Navanthe, S. B. (2000). Enabling scalable online personalization on the web. In *Proceedings of the 2nd acm conference on electronic commerce* (pp. 185–196). ACM.

Vieira, A. (2015). Predicting online user behaviour using deep learning algorithms. arXiv preprint, arXiv:1511.06247.

Wang, W., Yu, H., & Miao, C. (2017). Deep model for dropout prediction in MOOCs. In *Proceedings of the 2nd international conference on crowd science and engineering* (pp. 26–32). ACM.

Wu, F., Chiu, I.-H., & Lin, J.-R. (2005). Prediction of the intention of purchase of the user surfing on the web using hidden markov model. In *Services systems and services management, 2005. proceedings of ICSSSM'05. 2005 international conference on: 1* (pp. 387–390). IEEE.

Wu, Z., Tan, B. H., Duan, R., Liu, Y., & Mong Goh, R. S. (2015). Neural modeling of buying behaviour for e-commerce from clicking patterns. In *Proceedings of the 2015 international acm recommender systems challenge* (p. 12). ACM.

Xia, Y., Liu, C., Da, B., & Xie, F. (2018). A novel heterogeneous ensemble credit scoring model based on bstacking approach. *Expert Systems with Applications, 93*, 182–199.

Xing, Z., Pei, J., & Keogh, E. (2010). A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter, 12*(1), 40–48.

Zhao, Y., Yao, L., & Zhang, Y. (2016). Purchase prediction using tmall-specific features. *Concurrency and Computation: Practice and Experience, 28*(14), 3879–3894.