Scalable Data Debugging for Neighborhood-based Recommendation with Data Shapley Values

Barrie Kersbergen
Bol & University of Amsterdam
Amsterdam, The Netherlands
bkersbergen@bol.com

Olivier Sprangers
Nixtla
Amsterdam, The Netherlands
olivier@nixtla.io

Bojan Karlaš
Harvard University
Boston, United States
bkarlas@mgh.harvard.edu

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

Sebastian Schelter
BIFOLD & TU Berlin
Berlin, Germany
schelter@tu-berlin.de

Abstract

Machine learning-powered recommendation systems help users find items they like. Issues in the interaction data processed by these systems frequently lead to problems, e.g., to the accidental recommendation of low-quality products or dangerous items. Such data issues are hard to anticipate upfront, and are typically detected post-deployment after they have already impacted the user experience. We argue that a principled data debugging process is required during which human experts identify potentially hurtful data issues and preemptively mitigate them. Recent notions of "data importance," such as the Data Shapley Value (DSV), represent a promising direction to identify training data points likely to cause issues. However, the scale of real-world interaction datasets makes it infeasible to apply existing techniques to compute the DSV in recommendation scenarios.

We tackle this problem by introducing the KMC-Shapley algorithm for the scalable estimation of Data Shapley Values in neighborhood-based recommendation on sparse interaction data. We conduct an experimental evaluation of the efficiency and scalability of our algorithm on both public and proprietary datasets with millions of interactions, and showcase that the DSV identifies impactful data points for two recommendation tasks in e-commerce. Furthermore, we discuss applications of the DSV on real-world click and purchase data in e-commerce, such as identifying dangerous products or improving the ecological sustainability of product recommendations.

CCS Concepts

• **Information systems** → Recommender systems.

Keywords

Data importance, Data Shapley value, Neighborhood-based recommendation

ACM Reference Format:
Barrie Kersbergen, Olivier Sprangers, Bojan Karlaš, Maarten de Rijke, and Sebastian Schelter. 2025. Scalable Data Debugging for Neighborhood-based Recommendation with Data Shapley Values. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25), September 22–26, 2025, Prague, Czech Republic.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3705328.3748049

1 Introduction

Recommendation systems powered by machine learning help users to find the items that they need from large inventories. Real-world recommenders operate on large datasets that capture complex interactions between users and items. However issues in this interaction data frequently lead to system failures. Examples from the e-commerce domain include the accidental recommendation of dangerous items [28, 38], externally manipulated rankings [4], or third-party products with low-quality metadata [29, 37]. Furthermore, the data collection process is exposed to various sources of noise, e.g., "multi-journeys," where a single user shops for multiple things simultaneously (such as presents for family members of different age groups), or "unnatural interaction patterns" caused by bots crawling the website to record prices.

To make matters worse, these issues are hard to anticipate upfront, and are typically detected post-deployment after they have already impacted the user experience. For example, at Bol, a large European e-commerce platform, we recently became aware of a situation where sensitive adult items were included in the recommendations on product pages of children's toys, due to the unexpected co-occurrence of these types of products in some historical browsing sessions. This incident prompted an urgent live patch of the system with custom filtering rules, followed by the manual identification and removal of the session data in which these unwanted co-occurrences appeared.

Data debugging via data importance. Preventing such predicaments requires a *principled data debugging* process during which human experts identify potentially hurtful data issues and preemptively mitigate them. However, the scale of real-world training datasets renders this process prohibitively expensive and time-consuming. This shift reflects a growing awareness of the critical role of training data in shaping model behavior. A promising approach is to identify data points likely to cause issues via *data importance*, a concept recently introduced in the machine learning community. A popular notion of importance is the *Data Shapley value* (DSV) [8], which has been recognized for its effectiveness in some data debugging tasks [14, 15], including model auditing, dataset pruning, and outlier detection.

However, existing research on the DSV primarily focuses on classification tasks and has not been applied to recommendation tasks (except for a concurrent work on data pruning [42]). Moreover, current methods to overcome the exponential complexity of calculating the Data Shapley value are not suitable for large-scale recommender models (Section 2). Some techniques treat the model as a black box and incur repeated retraining [8, 19], which is infeasible for large datasets, while others make assumptions about the additivity of model quality metrics [13, 15] that do not apply to the ranking metrics of recommender systems.

Overview and contributions. Our work aims to bridge this gap. We focus on the large class of neighborhood-based recommendation models on sparse interaction data [3, 5, 6, 11, 12, 20, 21, 23, 25, 27, 32, 33]. We detail how to leverage the characteristics of such KNN models to efficiently compute Data Shapley values for datasets with millions of interactions (Section 3.2). Based on these characteristics, we design a specialized variant of a recent Monte Carlo-based algorithm [8] for estimating the DSV, which can skip certain expensive computations in many cases (Section 3.3). In particular, we provide the following contributions:

KMC-Shapley algorithm. We design the KMC-Shapley algorithm for the estimation of the Data Shapley value for neighborhood-based recommendation on sparse interaction data. This algorithm is a scalable variant of a recent Monte Carlo-based algorithm [8] for estimating the DSV, tailored to KNN models (Section 3).

Experimental evaluation. We conduct an experimental evaluation of the efficiency and scalability of our algorithm on both public and proprietary datasets with millions of interactions. Moreover, we showcase that the DSV identifies impactful data points for two recommendation tasks in e-commerce (Section 4).

Discussion of applications in e-commerce. We discuss applications of the DSV on click and purchase data in e-commerce from Bol, such as identifying dangerous and low-quality products as well as improving of the ecological sustainability of product recommendations via data pruning (Section 5).

Open source implementation. We implement our approach in Rust with a Python frontend and make it available at https://github.com/bkersbergen/illoominate.

2 Problem Statement

We formally introduce data importance (see Figure 1 inspired by [8]) and the scalability problem in the focus of this paper.

2.1 Data Importance

The goal of data importance is to quantify the impact of each individual training data point on the quality of a machine learning model [9]. Many notions of importance rely on measuring the impact of excluding a given data point from the model's training data (or certain subsets of it) [8, 19, 40]. Let $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ denote the set of n training data points whose importance we want to compute. Let the *utility function* $V(S)$ measure the performance of a model trained on a subset $S \subseteq \mathcal{D}$ of the training data. For example, V

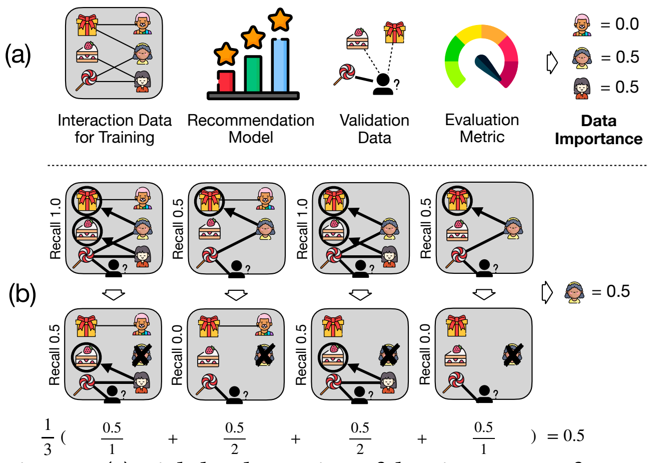


Figure 1: (a) High-level overview of data importance for recommender systems. (b) The importance of each piece of data (e.g., the interactions of a given user) is a weighted average of its contribution to each subset of the remaining interactions. The importance scores are determined by Equation (2).

might compute the recall of a recommendation model on held-out data $\mathcal{D}_{val}$. We discuss the two most prominent notions of data importance here:

Leave-One-Out error (LOO). The simplest way to measure the importance of a data point $\mathbf{x}_i \in \mathcal{D}$ is its leave-one-out error, i.e., the change in utility V when the data point $\mathbf{x}_i$ is excluded from the training data $\mathcal{D}$:

$$V(\mathcal{D}) - V(\mathcal{D} \setminus \{\mathbf{x}_i\}). \tag{1}$$

While the LOO is easy to compute, it is empirically found to be highly noisy [19] and typically outperformed by more complex notions of importance in its helpfulness for downstream tasks [9].

Data Shapley value (DSV). Recently, the Data Shapley value (DSV) has been proposed as an equitable way to measure the importance of training data points for a machine learning model [8]. The DSV determines the "value" ϕ_i of each data point $\mathbf{x}_i$ from the data $\mathcal{D}$ based on the well-known Shapley value from game theory [18]:

$$\phi_i = \frac{1}{n} \sum_{S \subseteq \mathcal{D} \setminus \{\mathbf{x}_i\}} \binom{n-1}{|S|}^{-1} V(S \cup \{\mathbf{x}_i\}) - V(S). \tag{2}$$

The DSV measures the weighted average of the marginal contribution $V(S \cup \{\mathbf{x}_i\}) - V(S)$ of adding the data point $\mathbf{x}_i$ to all $2^{|\mathcal{D}|-1}$ subsets S of the training data $\mathcal{D}$. See Figure 1(b) for a toy example. The DSV is challenging to compute but has been shown to work well empirically, e.g., for tasks like detecting mislabeled data [1, 8, 15, 19].

Note that a major advantage of data importance techniques for our scenario is that they are able to identify various kinds of erroneous data points without requiring explicit knowledge of the actual error types upfront, in contrast to classical supervised learning approaches.

2.2 Scalability Issues

Computing exact Data Shapley values, as defined in Equation (2), is intractable because the number of subsets to process is exponential in $|\mathcal{D}|$. This renders exact computation infeasible for real-world

Algorithm 1 TMC-Shapley algorithm as proposed in [8].

```
function TMC-SHAPLEY($\mathcal{D}, V$)
  $\phi = \{\phi_1, ..., \phi_n\} \leftarrow 0$
  while not converged :
    $\pi \leftarrow$ random permutation of data points in $\mathcal{D}$
    $v_{\text{prev}} \leftarrow V(\emptyset)$
    for $j \in 1 \dots n$ :                                   // Iterate through permutation
      if $|V(D) - v_{\text{prev}}| >$ performance tolerance          // Truncation
        $S^i_\pi \leftarrow$ set of data points before position $j$ in $\pi$
        $\mathbf{x}_i \leftarrow$ data point at position $j$ in $\pi$
        $v \leftarrow V(S^i_\pi \cup \{\mathbf{x}_i\})$                 // Retrain and evaluate model
        $\phi_i \leftarrow \phi_i + (v - v_{\text{prev}})$            // // Update marginal contribution
        $v_{\text{prev}} \leftarrow v$
  $t \leftarrow$ number of permutations evaluated
  $\phi \leftarrow \frac{1}{t}\phi$                                  // Normalise by number of iterations
  return Shapley values $\phi = \{\phi_1, \dots, \phi_n\}$
```

datasets with millions of interactions. Even worse, each subset S to process requires the re-training and evaluation of the underlying recommendation model.

TMC-Shapley. Computing the Data Shapley value ϕ_i for a data point $x_i \in \mathcal{D}$ can be formulated as an expectation calculation problem: $\phi_i = \mathbb{E}_{\pi \sim \Pi}\left[V(S^i_\pi \cup \{\mathbf{x}_i\}) - V(S^i_\pi)\right]$, where Π denotes the uniform distribution over all $n!$ permutations of the data points in $\mathcal{D}$ and S^i_π is the set of all data points coming before $\mathbf{x}_i$ in the permutation π [8]. For this formulation, Ghorbani and Zou [8] present the TMC-Shapley algorithm (see Algorithm 1), which conducts a Monte Carlo (MC) estimation of the Data Shapley values.[1] TMC-Shapley repeatedly samples a permutation π from Π, computes the marginal contribution $V(S^i_\pi \cup \{\mathbf{x}_i\}) - V(S^i_\pi)$ of a data point $\mathbf{x}_i$ over S^i_π in Lines 10 and 11, and iterates through all n data points in the permutation (Line 6). Furthermore, it applies a truncation technique, by stopping to compute marginal contributions once the obtained utility is within a threshold of the utility value $V(D)$ of the whole dataset (Line 7). A convergence test is performed by checking if the mean absolute percentage deviation of the Shapley value for all data points is below a given threshold.

Scalability issues in TMC-Shapley. Even Monte Carlo-based algorithms such as TMC-Shapley are difficult to scale to larger datasets [19], as the number of models to retrain and evaluate is linear in the number of data points in $\mathcal{D}$, which is still not feasible for large datasets, where a single training run can take multiple hours.

2.3 The Potential of KNN Models

Recent research details how to efficiently compute DSVs for KNN classifiers [13]. This direction is promising for recommendation scenarios as well, where KNN models have a long tradition, ranging from classical collaborative filtering [23, 32, 33] to recent state-of-the-art algorithms in session-based [6, 12, 25, 27], session-aware [20], next-basket [5, 11, 21], and within-basket recommendation [3], to applications for conversational recommendation [41]. However, existing approaches [13, 15] for efficiently computing DSVs for KNN classifiers are not directly applicable to KNN-based recommender systems. This is because they make several assumptions about the utility function. First, they assume that the model quality metric treats the contributions of the nearest neighbors independently, which is not the case for ranking-based metrics. Second, the computational efficiency bounds in [13, 15] are derived for classification tasks with a small number of class labels, which is not the case in recommender systems, which have to rank millions of items.

Research question. This leads us to the main research question of this paper: *Can we efficiently compute Data Shapley values for KNN-based recommenders on large sparse interaction datasets?*

[1]Note that we base Algorithm 1 on the authors' actual code from https://github.com/amiratag/DataShapley, which slightly differs from the formulation in their paper.

3 KMC-Shapley

Our goal is to design a general DSV estimation algorithm applicable to a wide range of recommendation models from the KNN family. In order to achieve this, we first define an abstract computational model capturing the general structure of KNN algorithms on sparse data in Section 3.1. Based on this, we design the KMC-Shapley algorithm, a specialized variant of TMC-Shapley (Algorithm 1), that skips expensive utility computations when the marginal contribution is known to be zero. We discuss the characteristics of KNN models that enable this in Section 3.2, and present the actual algorithm in Section 3.3.

3.1 Abstract Computational Model for Neighborhood-Based Recommendation

In the following, we discuss an abstract computational model that allows us to compute data importance in a general manner for KNN algorithms.

Sparse representation of interactions. Each set of interactions in the training data is encoded as a sparse representation $\mathbf{x}_i$. Thereby, the training data of observed interactions is turned into a retrieval corpus $\mathcal{D} = \{\mathbf{x}_i | \ i \in E\}$, where E denotes a use-case specific entity of interest (e.g., the ratings of a user, the ratings of an item, browsing sessions, shopping baskets, ...).

Retrieval-based inference. KNN models can be viewed as a combination of a retrieval function f_{ret} to query the retrieval corpus $\mathcal{D}$ and a prediction function f_{pred} to generate recommendations based on the retrieved representations. At inference time, KNN models compute recommendations for a query $\mathbf{x}_q$ in two stages:

(1) *Retrieval* – The k nearest neighbors $\mathbf{x}_{\alpha_1}, \dots, \mathbf{x}_{\alpha_k}$ for the queried representation $\mathbf{x}_q$ are retrieved via $f_{ret}(\mathbf{x}_q, \mathcal{D}, k)$, according to a model-specific ranking function.
(2) *Item scoring* – The top items to recommend are computed via the prediction function $f_{pred}(\mathbf{x}_q, \{\mathbf{x}_{\alpha_1}, \dots, \mathbf{x}_{\alpha_K}\})$, based on the query representation $\mathbf{x}_q$ and the retrieved neighbor representations $\mathbf{x}_{\alpha_1}, \dots, \mathbf{x}_{\alpha_k}$.

Note that this model is instantiated differently for different KNN algorithms. In classical item-based collaborative filtering [33], $\mathbf{x}_i$ represents all the ratings given to an item, in session-based recommendation such as VS-kNN [26], $\mathbf{x}_i$ represents a weighted sequence of clicks on items during a browsing session, and in next-basket recommendation algorithms such as TIFU-kNN [11], $\mathbf{x}_i$ consists of a time-weighted, aggregated representation of a user's shopping history.

Algorithm 2 KMC-Shapley algorithm.

```
function KMC-SHAPLEY($\mathcal{D}, V, k, \mathcal{D}_{\text{val}}$)
  $\phi = \{\phi_1, \ldots, \phi_n\} \leftarrow 0$                  // Initialize Data Shapley values
  $\text{N} \leftarrow \emptyset$                                  // Initialize neighbor index
  for $\mathbf{x}_q \in \mathcal{D}_{\text{val}}$ :                    // Populate index with neighbors
    $\text{N}_q = \{(\mathbf{x}_{\alpha_1}, s_{\alpha_1}), \ldots, (\mathbf{x}_{\alpha_M}, s_{\alpha_M})\} \leftarrow f_{\text{ret}}(\mathbf{x}_q, \mathcal{D})$
  while not converged :
    $\pi \leftarrow$ random permutation of data points in $\mathcal{D}$
    parfor $\mathbf{x}_q \in \mathcal{D}_{\text{val}}$      // Iterate over validation samples in parallel
      Sort $\text{N}_q$ according to the positions in $\pi$
      $\mathbf{h} \leftarrow$ min-heap of capacity $k$                 // Initialize min-heap
      Initialize pre-aggregate $\sigma_q$
      $v_{\text{prev}} \leftarrow 0$                              // Initialize previous utility
      for $(\mathbf{x}_{\alpha_i}, s_{\alpha_i}) \in \text{N}_q$ :
        if $|\mathbf{h}| < k$
          insert $(\mathbf{x}_{\alpha_i}, s_{\alpha_i})$ into $\mathbf{h}$
          Include $\mathbf{x}_{\alpha_i}$ into pre-aggregate $\sigma_q$
        else
          $(\mathbf{x}_h, s_h) \leftarrow$ data point and similarity of heap root
          if $s_{\alpha_i} > s_h$
            Remove $\mathbf{x}_h$ from pre-aggregate $\sigma_q$
            Include $\mathbf{x}_{\alpha_i}$ into pre-aggregate $\sigma_q$
            update heap root of $\mathbf{h}$ with $(\mathbf{x}_{\alpha_i}, s_{\alpha_i})$
        if $\mathbf{h}$ changed                // Set of k-nearest neighbors changed
          $v \leftarrow V_q(f_{\text{pred}}(\mathbf{x}_q, \sigma_q))$      // Utility with current neighbors
          $j \leftarrow$ position of $\mathbf{x}_{\alpha_i}$ in $\pi$
          $\phi_{\pi_j} \leftarrow \phi_{\pi_j} + (v - v_{\text{prev}})$            // Update Shapley value
          $v_{\text{prev}} \leftarrow v$                          // Update previous utility

  $t \leftarrow$ number of permutations evaluated
  $\phi \leftarrow \frac{1}{t}\phi$                          // Normalise by number of iterations
  return Data Shapley values $\phi = \{\phi_1, \ldots, \phi_n\}$
```

3.2 Data and Model Characteristics in Neighborhood-Based Recommendation

We discuss the data and model characteristics in neighborhood-based recommendation on sparse data, which allow us to skip the expensive utility computation in cases where we already know that the marginal contribution is zero.

Sparse retrieval. KNN algorithms typically operate on sparse interaction data between users and items. A common characteristic of these datasets is that they contain a wide selection of items, while each user only interacts with a tiny fraction of the available items. For example, the median session length in click datasets in e-commerce ranges from two to four clicks, even in scenarios with more than a million distinct items [17], and shopping baskets in purchase datasets contain less than ten items on average, even though there are tens of thousands of distinct items available [11]. As a consequence, the resulting interaction data is extremely sparse. To account for this, KNN algorithms use sparse representations for the interaction histories and apply a sparse retriever f_{ret} with a similarity function $sim(\mathbf{x}_q, \mathbf{x}_i)$ to rank a representation $\mathbf{x}_i$ in the retrieval corpus with respect to a query $\mathbf{x}_q$. These retrievers typically ignore pairs $(\mathbf{x}_q, \mathbf{x}_i)$ of representations with no overlap in item interactions, meaning that $\mathbf{x}_i$ will never be in the neighbor set of $\mathbf{x}_q$ in that case.

Locality. The prediction function f_{pred} only uses the k representations with the largest similarities returned by f_{ret}. Let $\Gamma_q(S)$ denote the k-th largest similarity score between the validation sample $\mathbf{x}_q$ and the representations from a set S. If $sim(\mathbf{x}_q, \mathbf{x}_i) < \Gamma_q(S)$, then adding $\mathbf{x}_i$ to S has no impact on the utility for $\mathbf{x}_q$. The sparsity of the interaction data and the locality property of KNN models allow us to work with the much smaller set of neighbors of each validation sample instead of having to process all data points from the training data in each iteration. Apart from being able to skip certain utility computations, we can further accelerate the algorithm based on the following characteristics.

Additivity of utilities. Analogous to classification and regression scenarios, the utility V of a model with respect to a validation dataset $\mathcal{D}_{val}$ in recommendation scenarios is computed by averaging the utilities for the individual validation samples $\mathbf{x}_q \in \mathcal{D}_{val}$ for common metrics like NDCG, MRR, Recall, etc. This additional property of V leads to $V(S^i_\pi \cup \{\mathbf{x}_i\}) - V(S^i_\pi) = \frac{1}{|\mathcal{D}_{val}|}\sum_{\mathbf{x}_q \in \mathcal{D}_{val}} V_q(S^i_\pi \cup \{\mathbf{x}_i\}) - V_q(S^i_\pi)$, where V_q computes the utility with respect to the validation sample $\mathbf{x}_q$. This enables a mapreduce-like parallelisation pattern, where we process each individual validation sample $\mathbf{x}_q \in \mathcal{D}_{\text{val}}$ in parallel and aggregate the resulting marginal contributions per training data point subsequently.

Linear aggregations at inference time. The prediction function f_{pred} in many KNN models first conducts a linear aggregation of the retrieved representations (e.g., a weighted sum of their sparse vector representations) and subsequently selects the items to recommend via a non-linear operation. Since the MC procedure requires us to sequentially scan the permutation of training points and investigate the impact of an additional sample (Lines 10–11 in Algorithm 1), it will repeatedly invoke f_{pred} with one new neighbor and $k-1$ neighbors that have been seen in the previous step. This makes it possible to reuse the aggregation result of the $k-1$ neighbors from the previous step. In order to achieve this, we make the KNN model maintain an aggregate σ_q of the current top-k neigbor set and directly compute predictions from this via $f_{pred}(\mathbf{x}_q, \sigma_q)$, which allows us to avoid repeating redundant aggregations.

3.3 KMC-Shapley Algorithm

Based on the outlined characteristics, we present *KMC-Shapley* in Algorithm 2. This variant of TMC-Shapley is tailored for KNN models and runs several orders of magnitude faster (as we experimentally show in Section 4.1.1).

KMC-Shapley starts with retrieving and indexing the top-M neighbors $\{\mathbf{x}_{\alpha_1}, \ldots, \mathbf{x}_{\alpha_M}\}$ of each validation sample $\mathbf{x}_q$ in Lines 4–5, where s_{α_i} denotes the similarity $sim(\mathbf{x}_q, \mathbf{x}_{\alpha_i})$. The parameter M denotes the maximum number of neighbors to consider per validation sample and is typically set to a high number. This parameter is inspired by recent statistics research which shows that high-cardinality subsets (with more than 100 elements) produce unreliable estimates of the marginal contribution in DSV computations [19]. Analogous to TMC-Shapley, our algorithm runs several MC iterations and generates a random permutation π of the data points in $\mathcal{D}$ in each iteration (Line 7). In contrast to TMC-Shapley, KMC-Shapley does not iterate over the data points in $\mathcal{D}$, but over each validation sample $\mathbf{x}_q$ in parallel (Line 8).

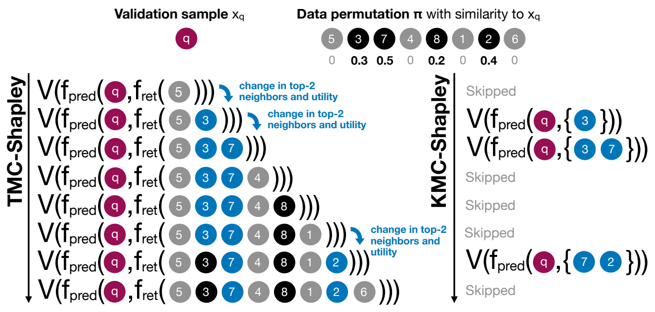

Validation sample x_q
Data permutation π with similarity to x_q
TMC-Shapley
KMC-Shapley
change in top-2 neighbors and utility
Skipped

Figure 2: Comparison of TMC-Shapley to KMC-Shapley for a toy example with eight data points, a single validation sample $\mathbf{x}_q$ and a model with $k = 2$. KMC-Shapley is able to skip many redundant computations, since only changes in the top-2 neighbor set of $\mathbf{x}_q$ lead to changes in utility. TMC-Shapley (left side) processes eight data subsets for the permutation π and evaluates the utility function V each time. KMC-Shapley (on the right side) takes the similarities of the data points to $\mathbf{x}_q$ into account and directly maintains the top-2 neighbor set of $\mathbf{x}_q$ (highlighted in blue). This allows KMC-Shapley to skip many redundant computations, since only changes in the top-2 neighbor set of $\mathbf{x}_q$ lead to changes in utility.

For each validation sample $\mathbf{x}_q$, our algorithm needs to consider its indexed neighbors only, according to their order in the permutation π (Line 9), as only the addition of new neighbors can produce a non-zero marginal contribution for $\mathbf{x}_q$. Our algorithm iterates through these neighbors (Line 13) and maintains the set of top-k neighbors in a binary heap (Lines 15 and 22). It simultaneously maintains the corresponding pre-aggregate σ_q of the top-k neigbor set under changes (Lines 16, 20 and 21). The computation of the marginal contribution of adding a new neighbor $\mathbf{x}_\alpha$ is only necessary (e.g., potentially non-zero) if the top-k neighbor set for $\mathbf{x}_q$ changes. In such cases, the change in utility is computed and used to update the DSV estimate corresponding to $\mathbf{x}_\alpha$ (Lines 23–27). Finally, the Data Shapley values are normalized by the number of iterations conducted and returned (Lines 29–31).

Complexity analysis. Let P denote the number of permutations considered. Algorithm 2 runs P iterations over $|\mathcal{D}_{\text{val}}|$ validation samples and has to process at most M neighbors per validation sample. Sorting these according to their positions in the permutation π can be done in $O(M \log M)$. For each neighbor, there is a cost of $\log k$ for a potential update of the heap and a constant cost of a single summation and subtraction for maintaining the pre-aggregate σ_q. Since $M \gg k$, this results in an overall time complexity of $O(P\,|\mathcal{D}_{\text{val}}|\,M \log M)$. These optimizations make KMC-Shapley significantly more scalable than the baseline, even for large validation sets and neighborhood sizes.

Toy example. We visualize the advantages of KMC-Shapley over TMC-Shapley on a toy example in Figure 2. For that, we show how both algorithms process a permutation π of a dataset set $\mathcal{D}$ with eight elements $\mathbf{x}_1, \ldots, \mathbf{x}_8$ for a KNN model with $k = 2$ and a single validation sample $\mathbf{x}_q$. The operations shown correspond to Lines 4–10 in Algorithm 1 for TMC-Shapley and to Lines 9–24 in Algorithm 2 for KMC-Shapley. This setup highlights how the sparsity and locality properties of kNN models reduce computation overhead, even in simple cases. On the left side, TMC-Shapley enumerates eight subsets of $\mathcal{D}$ according to the permutation π and evaluates the utility V of the resulting predictions for the validation sample $\mathbf{x}_q$ each time. KMC-Shapley (on the right side) takes the similarities of the data points in D to the validation sample $\mathbf{x}_q$ into account and directly maintains the top-2 neighbor set of $\mathbf{x}_q$ (highlighted in blue). This allows our algorithm to skip redundant computations, since only changes in the top-2 neighbor set of $\mathbf{x}_q$ will lead to changes in utility. Concretely, KMC-Shapley skips the utility computations for adding the data points $\mathbf{x}_5$, $\mathbf{x}_4$, $\mathbf{x}_1$ and $\mathbf{x}_6$, which have a similarity of zero to the validation sample $\mathbf{x}_q$. Moreover, KMC-Shapley can skip the utility computation for adding $\mathbf{x}_8$, since the similarity of $\mathbf{x}_8$ to $\mathbf{x}_q$ is too small to change the top-2 neighbor set. Overall, we see that KMC-Shapley can significantly reduce the number of required utility evaluations.

4 Evaluation

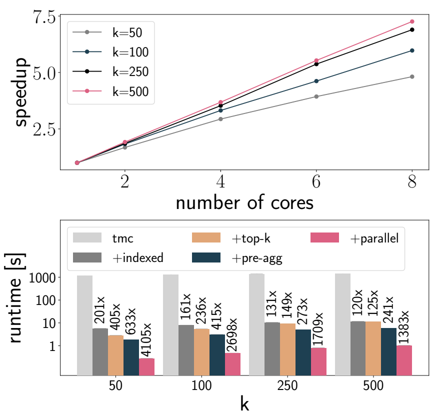
We evaluate the efficiency and scalability of KMC-Shapley and experimentally validate that it identifies impactful data points. We focus on two recommendation tasks in e-commerce, where KNN models provide competitive performance [21, 26]: Session-based recommendation [17, 26], where the goal is to predict the next item that a user will click on, and next-basket recommendation [3, 11, 21], where the goal is to predict the items in the next shopping basket purchased by a user. We run KNN-based recommenders in production for these tasks [16, 17, 39], which allows us to present experiments with real-world data and systems in this section.

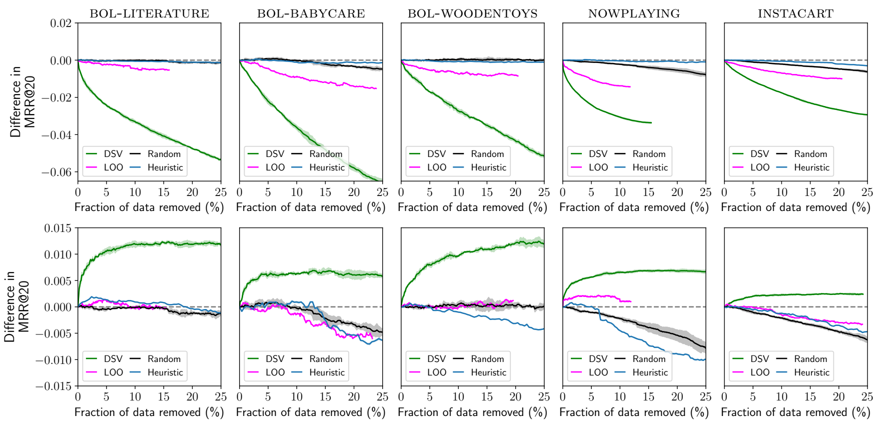
4.1 Efficiency & Scalability

4.1.1 Efficiency. The goal of our first experiment is to showcase that our proposed optimizations (Sections 3.2 & 3.3) for KMC-Shapley drastically reduce the runtime compared to TMC-Shapley.

Experimental setup. We run this experiment on a Windows 10 machine with an Intel i9-10900KF CPU. We use a sample of a public session dataset with 250,000 clicks as training data and 14,058 clicks as validation data, compute DSVs for session-based recommendation with a variant of the VS-kNN recommendation algorithm [17, 26], and repeat each run five times for $k \in \{50, 100, 250, 500\}$. We measure the mean runtime for a single MC iteration over all training data points with different algorithm variants.

We design this experiment as an ablation study for the proposed optimisations from originating from Section 3.3, based on the characteristics outlined in Section 3.2. For that, we introduce our proposed optimisations step by step to transform TMC-Shapley into KMC-Shapley. The baseline `tmc` denotes a Rust implementation of the vanilla TMC-Shapley algorithm (Algorithm 1), which retrains the recommendation model each time. Next, `+indexed` denotes a variant that does not retrain the KNN-SR model, but reuses the indexed neighbors per validation sample instead; the `+top-k` variant additionally maintains the top-k neighbor set in a binary heap and only computes marginal contributions once this set changes; the `+pre-agg` variant additionally maintains a pre-aggregate for accelerating inference and the final optimisation `+parallelism` parallelises the computation with eight threads, and corresponds to the fully optimised KMC-Shapley implementation.





(a) Efficiency and scalability of KMC-Shapley: its runtime scales linearly with the number of cores (top), and each optimization reduces the runtime (bottom).

(b) Impact of removing the most important data points (top row) and data points with negative importance scores (bottom row) from a session-based recommender system across datasets. Data identified by the Data Shapley value (DSV) has stronger impact than data identified by leave-one-out error (LOO), random removal, or session length as a simple heuristic.

Figure 3: Evaluation of KMC-Shapley for session-based recommendation: our algorithm is efficient and scales linearly with the number of cores (left); data identified by the Data Shapley value has the strongest impact on model performance (right).

Results and discussion. We plot the resulting mean runtimes (and the speedup over the `tmc` baseline) on a logarithmic scale in the bottom of Figure 3a. We observe that all our performance optimisations are beneficial, as each optimisation further reduces the runtime. In this experiment, KMC-Shapley is three orders of magnitude faster than the vanilla TMC-Shapley implementation which repeatedly retrains the model. As expected, the largest runtime reduction originates from reusing the neighbor sets and avoiding model retraining in the `+indexed` variant (which exploits the "sparse retrieval" and "locality" characteristics outlined in Section 3.2). In this experiment, we find that KMC-Shapley can conduct a single iteration for all training data points in a second or less for all values of k. The results confirm that it is indeed possible to design customizsed, highly accelerated variants of the Data Shapley value computation for nearest neighbor models on sparse data.

4.1.2 Scalability. The goal of the next experiment is to show that our implementation benefits from additional computational resources, such as CPU cores.

Experimental setup. We experiment with a session-based recommender using a variant of the VS-kNN algorithm [17, 26] on a large sample of 10,431,353 clicks in 1,668,295 sessions from Bol. We run KMC-Shapley on this data with a validation set containing 250,000 clicks in 40,023 sessions, vary the number of neighbors k, and increase the number of cores from one to eight. We execute this experiment on a machine with a Mac M1 Pro, repeat each run six times, and report the speedup over the single-threaded baseline.

Results and discussion. We plot the resulting speedups on the top of Figure 3a. We observe that the runtime scales linearly with the number of available cores, which is expected due to the map-reduce-like computational pattern in KMC-Shapley (Section 3.2). We observe diminishing returns for smaller values of k, which we attribute to the fact that the algorithm is less dominated by the computational efforts for inference in these cases. Even on this large dataset of over 10 million clicks, a single KMC iteration with eight cores only takes 136 seconds on average for all training data points.

4.2 Impact of Data Removal

Next, we evaluate how well KMC-Shapley identifies impactful data points for interaction data. For that, we adopt a common data removal experiment [8, 15, 19] to our recommendation setup. In this experiment, data points are removed from the training data according to a given order (e.g., sorted ascendingly or descendingly with respect to their importances), and the prediction quality of a model trained on the remaining data is repeatedly evaluated on a held-out test set. The rate at which the prediction quality changes indicates how impactful the chosen removal order is.

Experimental setup. We conduct this data removal experiment for session-based recommendation [6, 26]. Concretely, we use a variant of the VS-kNN recommendation algorithm [17, 26] with hyperparameters $k = 50$ and $M = 500$. We experiment with samples from two public datasets (NOWPLAYING, INSTACART) and three proprietary click datasets (BOL-LITERATURE, BOL-BABYCARE, BOL-WOODENTOYS) with samples of up to 1.7 million clicks from different product categories on Bol. We prepare the datasets as follows: we conduct a temporal split of the sessions into training sessions and held-out sessions, and we randomly split the held-out sessions into 50% validation and 50% test data. We make sure that we only retain sessions with items seen in the training data (e.g., we ignore cold-start items for which no clicks have been seen yet). Table 1 summarises the statistics of the resulting datasets.

Note that it is crucial to choose a large enough validation set to avoid a high variance in the DSV scores; this can be determined upfront via preliminary experiments with differently sized validation sets and different random seeds.

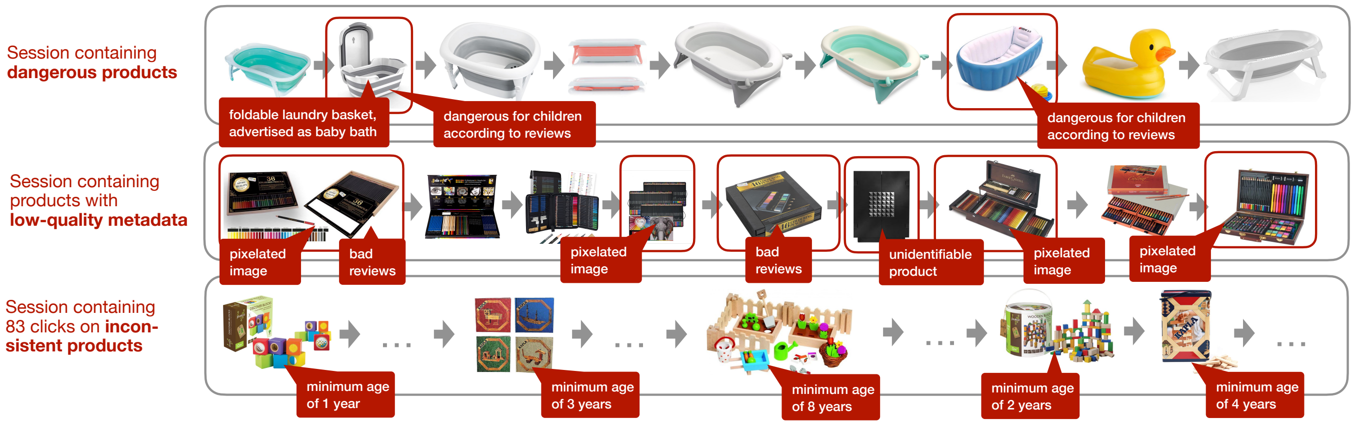

Session containing dangerous products
foldable laundry basket, advertised as baby bath
dangerous for children according to reviews
dangerous for children according to reviews
Session containing products with low-quality metadata
pixelated image
bad reviews
pixelated image
bad reviews
unidentifiable product
pixelated image
pixelated image
Session containing 83 clicks on inconsistent products
minimum age of 1 year
minimum age of 3 years
minimum age of 8 years
minimum age of 2 years
minimum age of 4 years

Figure 4: Examples of sessions with negative importance scores in Bol's click data (*each box represents a single session consisting of a sequence of clicks on items, arrows show the order of clicks*). We encounter sessions containing dangerous products (e.g., a foldable laundry basket incorrectly advertised as baby bath), sessions with metadata quality issues (e.g., pixelated and unidentifiable images) as well as sessions with inconsistent products (wooden toys for different age ranges).

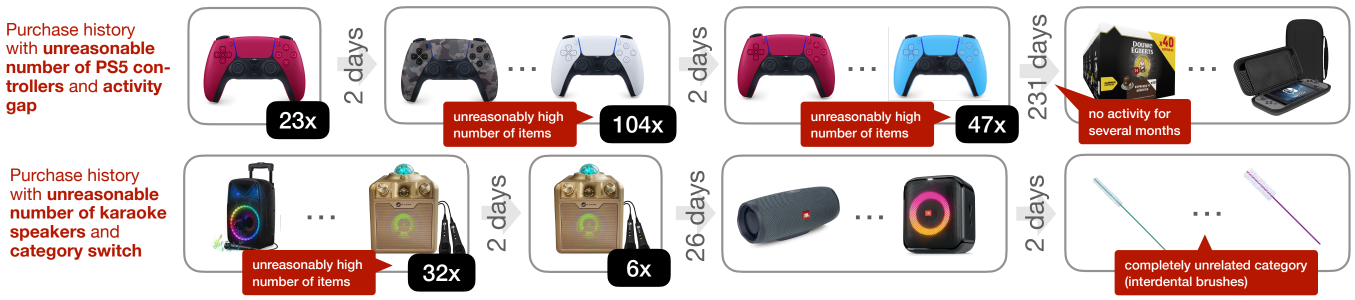

Purchase history with unreasonable number of PS5 controllers and activity gap
23x
2 days
unreasonably high number of items
104x
2 days
unreasonably high number of items
47x
231 days
no activity for several months
Purchase history with unreasonable number of karaoke speakers and category switch
unreasonably high number of items
32x
2 days
6x
26 days
2 days
completely unrelated category (interdental brushes)

Figure 5: Examples of purchase histories with negative importance scores from Bol (*boxes indicate items bought together in a shopping cart, arrows point to the next shopping cart*). The histories contain electronics items bought in unreasonably high numbers (e.g., over a hundred PlayStation controllers), an activity gap of more than half a year, and switch to unrelated product categories (coffee, dental products) at some point.

Table 1: Datasets for session-based recommendation.

				#clicks		
Dataset		**#sessions**	**#items**	train	valid	test
BOL-LITERATURE	proprietary	393,655	50,202	1,704,661	168,109	166,289
BOL-BABYCARE	proprietary	29,324	3,161	127,088	13,351	13,395
BOL-WOODENTOYS	proprietary	98,420	8,937	437,329	44,427	44,825
NOWPLAYING	public	97,922	196,531	489,367	58,277	57,616
INSTACART	public	21,068	18,661	124,376	60,104	59,363

We compute Data Shapley values via KMC-Shapley and LOO errors with respect to the MRR for the first 20 recommended items (referred to as MRR@20). We evaluate the impact of removing highly important sessions with positive DSVs in descending order. We repeat this analogously for sessions with positive leave-one-out errors and also include two baselines, one which simply removes data points at random, and another heuristic baseline which considers the number of clicks contained in a session (based on the assumption that very long sessions are likely to originate from bots and crawlers). We replicate this experiment for removing low-scored data, where we remove the sessions with negative DSVs and LOOs in ascending order instead.

Results and discussion. Figure 3b shows the absolute differences in MRR@20 on the held-out test data when removing up to 25% of the training sessions. Each experiment is repeated three times, and we report mean values as solid lines and standard deviations as shaded areas. When removing highly important data, we observe clear differences across the methods. Removing sessions ranked by Data Shapley values (DSV) causes the steepest decline in MRR@20, up to 0.06, demonstrating that KMC-Shapley more effectively identifies high-impact training data than LOO, random removal or removal based on the heuristic. When removing sessions with negative importance scores, DSV again yields the largest improvements across datasets, confirming that these sessions often represent harmful or noisy data. LOO-based removal improves performance less consistently, and removal based on random choice shows only minor effects in either direction. Removing up to 5% of the longest sessions slightly outperforms random removal, but performance degrades beyond that. This highlights the limited effectiveness of rule-based heuristics compared to learned data importance scores. We now evaluate scalability on real-world datasets. For this experiment, we use a Mac M3 Pro with 6 cores, on which the DSV computation requires between 1,420 to 2,481 iterations to converge and the time per iteration varies from 0.7 seconds on BOL-BABYCARE to

21.4s on INSTACART. Even for the largeset dataset BOL-LITERATURE with more than two million clicks, the computation finishes in under 10 hours. We run another experiment with a dataset of over two million product purchases and a next-basket recommendation model [11] from our e-commerce platform. The results observed are in line with the results from the previous experiment on session-based recommendation. Due to lack of space, we refer to our code repository for details on the experimental setup and results.

5 Applications

We discuss some applications of data importance at Bol.

Identifying outliers and corrupted data. The main purpose of data importance is to help us find corrupted and harmful interaction data in the production recommender system [17] of Bol. This session-based recommender serves the "others also viewed" recommendations on the product pages. Apart from improving our recommender systems, uncovering data issues is also important for other teams, e.g., those responsible for product quality and risk issues on the platform.

In order to showcase this use case, we draw a large sample of historical click data from the babycare, wooden toys, and wooden pencils categories, and compute DSVs with MRR@20 as the target metric of a variant of our VS-kNN recommendation algorithm [17, 26]. We find that the lowest-scored sessions contain inconsistencies, problematic products, and corrupted data. We show three example sessions taken from the ten lowest-scored sessions found in these data samples in Figure 4. All of these sessions are harmful to the recommender system (i.e., including them in the training data negatively impacts prediction quality) and suffer from data issues. The top-most session originates from a user exploring baby bath products and for example contains a foldable laundry basket, which has been incorrectly advertised as a baby bath by its seller, and which according to its reviews is actually dangerous to use for bathing children. The remaining two sessions shown in the figure suffer from different issues. The second session contains pencil products, many of which suffer from quality issues in their metadata, such as pixelated and unidentifiable images. The third session contains a massive number of 83 clicks on wooden toys, and the toys target highly variable age ranges (one-year-olds to eight-year-olds), making this session unsuitable as a basis for recommendation. These examples showcase that DSVs identify low-quality interaction data with issues that are hard to anticipate upfront. We repeat this analysis for a large sample of purchase histories from Bol and a next-basket recommendation model [11] (a variant of which we run in production as well), leveraging NDCG@20 as the target metric. We again find that the lowest-scored purchase histories contain data that is not helpful for our recommender systems, and refer to Figure 5 for examples. The histories contain electronics items bought in unreasonable numbers (e.g., over a hundred PlayStation controllers), switch to completely unrelated product categories (coffee, dental products) at some point, and are therefore clearly uninformative for a recommender system. We attribute such purchase histories to commercial accounts, which buy large numbers of products for several thousands of Euros as a response to price promotions, probably intended for reselling.

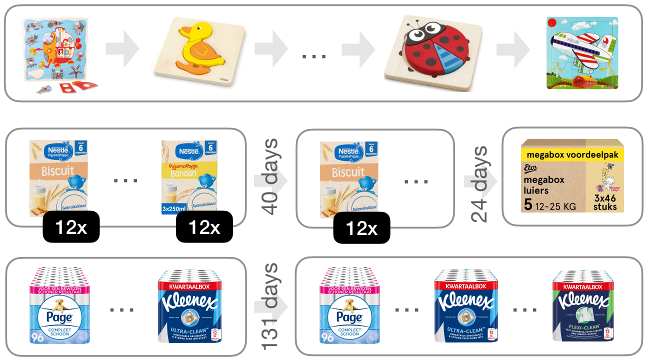


Figure 6: Examples of high-value sessions and purchase histories with consistent selections and high turn-over items.

We also investigate the sessions and purchase histories with the highest DSVs from this experiment and plot examples in Figure 6. We find that the DSVs identify high-value sessions with consistent product selections in click data, and high-value shopping baskets with high turnover items (baby formula, toilet paper) in purchase history data.

Increasing the sustainability of recommendations via data pruning. A major goal of Bol is to make sustainable shopping easier for customers. We present an experimental application of DSVs, which contributes to this mission. The aim here is to increase the number of sustainable items in the predictions of our session-based recommender, without compromising the prediction quality. We choose a data-centric approach that prunes the underlying data in an effort to remove the histories containing unsustainable products. This allows us to maintain the prediction quality of our recommender system by letting the real histories with a higher number of sustainable products serve as a basis for future recommendations. To modify the behavior of our recommender system through interventions on the training data, we need a utility function able to measure the desired behavior. To this end, we augment the product metadata with a binary flag that indicates whether an item was produced in a sustainable manner. Furthermore, we modify the MRR metric to include a sustainability term that expresses the number of sustainable products in a given recommendation. Specifically, we define a utility function, which we call *SustainableMRR@t* as $0.8 \cdot MRR@t + 0.2 \cdot \frac{s}{t}$. This utility combines the MRR@t with a sustainability coverage term $\frac{s}{t}$, where s denotes the number of sustainable items among the t recommended items.

For this internal experiment, we compute Data Shapley values and leave-one-out errors for our recommender system with SustainableMRR@21 as a utility on click datasets from the baby care and wooden toys categories on Bol, which contain large numbers of sustainable products. Next, we prune the training data based on the computed DSVs (by removing the 5% lowest scored data points) and evaluate the recommendations on held-out test data. We observe that removing this data significantly increases the SustainableMRR@21 metric across all cases and that both the MRR@21 as well as the sustainability coverage increase as a result of the pruning. Pruning based on LOO leads to unreliable results in our experiment since we observe a small increase for the wooden toys category, but a decrease in SustainableMRR@21 for recommendations of baby care items. In summary, these results confirm that we can leverage DSVs with custom-designed utility functions for the data-centric optimization of existing recommendation models.

6 Related Work

KNN-based recommender systems. KNN models have a long tradition in recommender systems, ranging from classical collaborative filtering [23, 32, 33] to recent state-of-the-art algorithms in session-based [6, 12, 25, 27], session-aware [20], next-basket [5, 11, 21], and within-basket recommendation [3], to applications for conversational recommendation [41].

Error detection for ML data. Detecting errors in data is a core research direction in data management [2, 22], and several approaches tailored to ML applications have been proposed. Google TFX [30] and Deequ [34, 35] infer integrity constraints for ML data based on data profiling, and follow-up approaches learn to validate ML data from historically observed data partitions [31, 36], few-shot examples [10] or via self-supervised learning [24].

Novelty. To the best of our knowledge, we are the first to explore data importance for KNN-based recommenders. The only other application of the DSV in recommender systems (that we are aware of) is the selection of data points for pruning [42], where it successfully uncovers errors introduced by synthetic random noise.

7 Conclusion

We have presented a scalable algorithm for debugging the interaction data of KNN-based recommender systems via the Data Shapley value, and showcased that it identifies impactful data points in datasets with millions of interactions.

Limitations. A limitation of our approach is that the proposed optimizations only benefit nearest neighbor models on sparse interaction data, but are not directly applicable to neural models. This is due to the fact that several of the outlined properties from Section 3.2 do not hold for these models, as they operate on dense representations and lack localised influence. Moreover, an open challenge is how to efficiently maintain data importance scores incrementally as new interactions arrive. This is especially important for growing real-world datasets, where one would like to avoid having to rerun the DSV computation from scratch for data updates.

Future work. In the future, we plan to include additional recommendation models [3, 5, 7] and data importance algorithms [19, 40] in our implementation. Furthermore, we will explore how well the importance scores from KNN models serve as proxies for data importance in computationally expensive neural models.

Acknowledgments

This research was supported by Ahold Delhaize, through AIRLab Amsterdam, the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.1389.20.183, and KICH3.LTP.20.006, and the European Union's Horizon Europe program under grant agreement No 101070212. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

[1] 2023. Proactively Screening Machine Learning Pipelines with ArgusEyes. *Schelter, Sebastian and Grafberger, Stefan and Guha, Shubha and Karlas, Bojan and Zhang, Ce* (2023).
[2] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting Data Errors: Where Are We and What Needs to Be Done? *VLDB* (2016).
[3] Mozhdeh Ariannezhad, Ming Li, Sebastian Schelter, and Maarten de Rijke. 2023. A Personalized Neighborhood-based Model for Within-Basket Recommendation in Grocery Shopping. *WSDM* (2023).
[4] Buzzfeed. 2019. "Amazon's Choice" Does Not Necessarily Mean A Product Is Good. https://www.buzzfeednews.com/article/nicolenguyen/amazons-choice-bad-products
[5] Guglielmo Faggioli, Mirko Polato, and Fabio Aiolli. 2020. Recency Aware Collaborative Filtering for Next Basket Recommendation. *UMAP* (2020).
[6] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. *RecSys* (2019).
[7] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2019. Sequence and Time Aware Neighborhood for Session-based Recommendations: Stan. *SIGIR* (2019).
[8] Amirata Ghorbani and James Zou. 2019. Data Shapley: Equitable Valuation of Data for Machine Learning. *ICML* (2019).
[9] Zayd Hammoudeh and Daniel Lowd. 2024. Training Data Influence Analysis and Estimation: A Survey. *Machine Learning* (2024).
[10] Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. 2019. Holodetect: Few-shot Learning for Error Detection. *SIGMOD* (2019).
[11] Haoji Hu, Xiangnan He, Jinyang Gao, and Zhi-Li Zhang. 2020. Modeling Personalized Item Frequency Information for Next-basket Recommendation. *SIGIR* (2020).
[12] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks Meet the Neighborhood for Session-based Recommendation. *RecSys* (2017).
[13] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. 2019. Efficient Task-specific Data Valuation for Nearest Neighbor Algorithms. *VLDB* (2019).
[14] Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. 2021. Scalability vs. Utility: Do We Have to Sacrifice One for the Other in Data Importance Quantification? *CVPR* (2021).
[15] Bojan Karlaš, David Dao, Matteo Interlandi, Sebastian Schelter, Wentao Wu, and Ce Zhang. 2023. Data Debugging with Shapley Importance over Machine Learning Pipelines. *ICLR* (2023).
[16] Barrie Kersbergen and Sebastian Schelter. 2021. Learnings from a retail recommendation system on billions of interactions at bol.com. *ICDE* (2021).
[17] Barrie Kersbergen, Olivier Sprangers, and Sebastian Schelter. 2022. Serenade-low-latency session-based recommendation in e-commerce at scale. *SIGMOD* (2022).
[18] Harold William Kuhn and Albert William Tucker. 1953. *Contributions to the Theory of Games.* Number 28. Princeton University Press.
[19] Yongchan Kwon and James Zou. 2022. Beta Shapley: A Unified and Noise-Reduced Data Valuation Framework for Machine Learning. *AISTATS* (2022).
[20] Sara Latifi, Noemi Mauro, and Dietmar Jannach. 2021. Session-aware Recommendation: A Surprising Quest for the State-of-the-art. *Information Sciences* (2021).
[21] Ming Li, Sami Jullien, Mozhdeh Ariannezhad, and Maarten de Rijke. 2023. A Next Basket Recommendation Reality Check. *ACM TOIS* (2023).
[22] Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. 2021. CleanML: A Study for Evaluating the Impact of Data Cleaning on ML Classification Tasks. *ICDE* (2021).
[23] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com Recommendations: Item-to-item Collaborative Filtering. *IEEE Internet Computing* (2003).
[24] Zifan Liu, Zhechun Zhou, and Theodoros Rekatsinas. 2022. Picket: Guarding against Corrupted Data in Tabular Data during Learning and Inference. *VLDB Journal* (2022).
[25] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of Session-based Recommendation Algorithms. *UMAP* (2018).
[26] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Performance Comparison of Neural and Non-neural Approaches to Session-based Recommendation. *RecSys* (2019).
[27] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2021. Empirical Analysis of Session-based Recommendation Algorithms. *UMAP* (2021).
[28] Channel 4 News. 2017. Potentially Deadly Bomb Ingredients are 'Frequently Bought Together' on Amazon. https://www.channel4.com/news/potentially-deadly-bomb-ingredients-on-amazon
[29] Vice News. 2023. AI-Generated Books of Nonsense Are All Over Amazon's Bestseller Lists. https://www.vice.com/en/article/ai-generated-books-of-nonsense-are-all-over-amazons-bestseller-lists/
[30] Neoklis Polyzotis, Martin Zinkevich, Sudip Roy, Eric Breck, and Steven Whang. 2019. Data Validation for Machine Learning. *MLSys* (2019).

[31] Sergey Redyuk, Zoi Kaoudi, Volker Markl, and Sebastian Schelter. 2021. Automating Data Quality Validation for Dynamic Data Ingestion. *EDBT* (2021).

[32] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. Grouplens: An Open Architecture for Collaborative Filtering of Netnews. *CSCW* (1994).

[33] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. *WWW* (2001).

[34] Sebastian Schelter, Stefan Grafberger, Philipp Schmidt, Tammo Rukat, Mario Kiessling, Andrey Taptunov, Felix Biessmann, and Dustin Lange. 2019. Differential Data Quality Verification on Partitioned Data. *ICDE* (2019).

[35] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. 2018. Automating Large-Scale Data Qquality Verification. *VLDB* (2018).

[36] Shreya Shankar, Labib Fawaz, Karl Gyllstrom, and Aditya G. Parameswaran. 2023. Moving Fast with Broken Data. *CIKM* (2023).

[37] Ars Technica. 2024. Lazy Use of AI Leads to Amazon Products Called "I Cannot Fulfill that Request". https://arstechnica.com/ai/2024/01/lazy-use-of-ai-leads-to-amazon-products-called-i-cannot-fulfill-that-request/

[38] New York Times. 2022. Lawmakers Press Amazon on Sales of Chemical Used in Suicides. https://www.nytimes.com/2022/02/04/technology/amazon-suicide-poison-preservative.html

[39] Maria Vechtomova et al. 2023. Databricks AI Summit: Streamlining API Deploy ML Models Across Multiple Brands: Ahold Delhaize's Experience on Serverless. Retrieved July 5, 2024 from https://www.youtube.com/watch?v=GSJFyoBiCXk

[40] Jiachen T. Wang and Ruoxi Jia. 2023. Data Banzhaf: A Robust Data Valuation Framework for Machine Learning. *AISTATS* (2023).

[41] Zhouhang Xie, Junda Wu, Hyunsik Jeon, Zhankui He, Harald Steck, Rahul Jha, Dawen Liang, Nathan Kallus, and Julian McAuley. 2024. Neighborhood-Based Collaborative Filtering for Conversational Recommendation. *RecSys* (2024).

[42] Yansen Zhang, Xiaokun Zhang, Ziqiang Cui, and Chen Ma. 2025. Shapley Value-driven Data Pruning for Recommender Systems. *KDD* (2025).