NUDGING TOWARDS SUSTAINABLE CHOICES VIA RECOMMENDER SYSTEMS

NUDGING TOWARDS SUSTAINABLE CHOICES VIA RECOMMENDER SYSTEMS

to obtain the degree of Master in Computer Science with Specialization in Artificial Intelligence Technology at Delft University of Technology.

by

Raoul KALISVAART

Born in The Hague, The Netherlands.

Multimedia Computing Group, Faculty of Electrical Engineering, Mathematics and Computer Science (Faculteit Elektrotechniek, Wiskunde en Informatica), Delft University of Technology, Delft, The Netherlands.

Thesis committee

Chair,	Dr. Ir. Odette Scharenborg, Faculty EEMCS, TU Delft
Daily Supervisor:	Dr. Elvin Isufi, Faculty EEMCS, TU Delft
Member:	Dr. Luis Miranda da Cruz, Faculty EEMCS, TU Delft

Keywords: Recommender Systems, Nudging, Machine Learning for Climate Change

SUMMARY

We all know the possible consequences of global warming, rising temperatures, flooded cities and destroyed ecosystems. One of the causes is the emission of gases, predominantly CO_2, which is increased by the growing E-commerce market. E-commerce companies rely on recommender systems to stimulate users to purchase products. We are convinced that we can use the core strength of recommender systems, influencing decision making, to steer users towards eco-friendly choices. Therefore, in this thesis, we research how greenness can be integrated into recommender systems. We present the first recommender system dataset that includes greenness, we benchmark several recommendation algorithms and we propose a strategy to increase recommendation greenness. To create the dataset, we annotate an existing recipe recommendation dataset with recipe greenness. For our benchmarking experiment, we propose metrics to measure recommendation greenness, which we use to show that no recommendation algorithm is fundamentally greener than others. Lastly, we propose a re-ranking method for improving the greenness of recommendation rankings. We use the method to explore the trade-off between accuracy and greenness and we show that it is possible improve the greenness of recommender systems significantly with little loss of accuracy.

ACKNOWLEDGEMENTS

I would like to thank all the people that made this journey possible. First of all, I want to thank my supervisor, Elvin Isufi, for his supervision throughout the entire thesis. I learned much from our in-depth discussions about virtually every aspect of research, but above all, I enjoyed them. Thank you for all your energy and feedback. It was a good learning experience. I would also like to acknowledge Odette Scharenborg and Luís Cruz for accepting to be part of my thesis committee.

Moreover, I want to thank my family for supporting me throughout the months of hard working. I especially want to thank my girlfriend, Anoeya, for being there for me to support and help me with every step along the way.

CONTENTS

1

INTRODUCTION

Rising temperatures, flooded cities, destroyed ecosystems and heatwaves will likely be our and earth's future if we let global warming run its course. The emission of gasses, predominantly CO_2, can partly be ascribed to reinforcing global warming, and human activity has skyrocketed this throughout the years [1]. A factor in this phenomenon is the increasing market size of e-commerce, which grew by 7% since 2019 [2]. This sector engages in product selling via digital means. While e-commerce provides an easy way to purchase products and can save consumers' time, its growth is also responsible for negatively impacting the environment, e.g. by increasing emissions and pollution. For example, the footprint of the e-commerce giant Amazon was 71.54 million metric tons CO_2 equivalent in 2021 [3], which is only slightly less than the 78 million metric tons CO_2 equivalent of Paris, the capital of France [4].

We believe that, to challenge this, the starting point is at the level of the consumer. After all, the consumption behaviour of these individuals is what drives the ecommerce sells. If we can influence consumers' decision towards being more conscious about the environment, this could have a positive influence on the emissions of ecommerce. Influencing decision-making is exactly the core strength of recommender systems. By turning data on user preferences into accurate recommendations, they assist users in making choices. Recommender systems have been proven effective in turning large corpora of product data into concrete recommendations, improving user-satisfaction and sales [5]–[7]. At the same time, recommender systems are already widely utilized by ecommerce companies. For example, it is estimated that 35% of Amazon's sales can be attributed to recommender systems [8].

So, if recommender systems can, succesfully, steer users' decisions towards more preferable options, why should we not make them more green? There is plenty literature about comparing and improving recommender systems, but there are no works that thoroughly investigate how to include greenness in recommender systems. We are convinced that recommender systems are ideal for achieving this task, as they can implicitly stimulate, or *nudge*, users to make more responsible decisions, without requiring them to change their behavior. Therefore, we present the first in-depth research on modifying

1

recommender systems to generate green recommendations.

In this thesis, our main research question is: 'To what extent can greenness be integrated into recommender systems?'. To cover all grounds we answer the following subquestions:

(SQ1): Can we make a dataset to evaluate the greenness of recommender systems?

(SQ2): Is there a difference in the greenness of current recommender system algorithms?

(SQ3): How can we increase the greenness of recommendations?

Firstly, to have a dataset that is able to evaluate the greenness of recommender systems, we develop the first of its kind, namely the RecipeEmissions dataset. We annotate a recipe recommendation dataset [9] with data on ingredient quantities and CO_2 emissions using manual and automated techniques. To validate the dataset we do a comparison with several often-used recommender system datasets.

Secondly, to evaluate the difference in greenness of current recommender systems algorithms, we develop two metrics, one for rating-based and one for ranking-based recommender systems. We benchmark the greenness of several common algorithms: the global average, user-based k-nn, item-based k-nn, SVD, SVD++, co-clustering and SLIM. We compare their greenness and set a baseline to improve upon.

Thirdly, in testing if the greenness of recommendations can be increased, we propose a method of re-ranking recommendations. In the process of making these recommendations as green as possible, we also focus on its effect on the recommendations' accuracy.

This thesis is organized as follows. First, Chapter 2 presents the background information. Second, Chapter 3 investigates the literature that relates to our work. Third, In Chapter 4 we develop the RecipeEmissions dataset. Next, Chapter 5 evaluates the greenness of commonly-used recommender systems. Chapter 6 proposes a strategy for improving the greenness of recommendations. Lastly, Chapter 7 concludes our work and provides recommendations for future research.

2

BACKGROUND

In this chapter, we present the background knowledge that will be utilized in this thesis. Section 2.1 explains recommender systems. Section 2.2 explains how recommender systems can be evaluated, both in terms of their accuracy as well as other objectives. Section 2.3 concludes this chapter.

2.1. RECOMMENDER SYSTEMS

People heavily rely on recommendations of others; whether someone intends to find a movie to watch, a restaurant to eat at or a destination to travel to. Recommender systems give an automated alternative for this [10]. They namely provide the most relevant items for users from large amounts of data [11]. In practice, this a prediction task: given a set of users and a set of items, recommender systems predict, for each user-item pair, how relevant the specific item would be to the specific user. The predictions are usually based on known user-preference data. Such information can be either explicit, by collecting ratings and likes, or implicit, by inferring them from behaviour. Examples of implicit preference are monitoring website visits and the clicking activity of users [12]. When a user rates an item (either explicitly or implicitly), it is generally referred to as an 'interaction'.

There exist many different types of recommender systems. Roughly, they can be categorized according to their prediction task. We cover this categorization, which is done between rating- and ranking-based recommender systems.

2.1.1. RATINGS AND RANKINGS

Rating-based. The recommendation task can be rephrased as predicting the ratings that users would give to items [13]. To achieve this, rating-based recommender systems learn from known user ratings. Equation 2.1 formalizes this concept. After ratings have been predicted, recommendations can be found by finding the items with the highest predictions.

$$\mathbf{R} : \mathbf{users} \times \mathbf{items} \rightarrow \mathbf{ratings} \tag{2.1}$$

Ranking-based. Differently from rating-based ones, ranking based systems do not predict numerical ratings. Instead, they predict rankings of recommendations: lists that are ordered on the predicted utility for the user. Ranking-based recommender systems are useful in scenarios where users only consider a limited number of recommendations. In those cases, it is less important how well the ratings of low-ranking interactions can be predicted. An often-used method to generate rankings is to order these items based on their predicted rating. There are, however, also methods that directly order recommendations, by comparing pairs of items to predict which would be preferred over the other [14].

2.1.2. RECOMMENDER SYSTEM ALGORITHMS

There are countless recommender systems available, all having their benefits and challenges. We explain several commonly-used algorithms that we will use later on.

Global-mean. This method entails the simplistic strategy of using the global mean of all ratings in a training set as the prediction for every interaction in the test set. This is a simple method and it is often used as a baseline to cast the improvements of more sophisticated methods.

User/item-knn [5]. Nearest neighbour algorithms use similarities between users and items to predict ratings. For every user-item interaction it needs to predict a rating for, a user k-nearest neighbour algorithm selects the k most similar users (to the user it needs to predict for) and computes a weighted average for the ratings that these users assigned to the item. An item-knn algorithm works similarly, but, instead it finds k similar items. For a user u, an item i, a set of most similar users (neighbours) N of size k and a similarity measure $sim(u,v)$, Equation 2.2 defines user-based knn prediction for rating $\hat{r}_{ui}$. An often-used similarity measures is the cosine similarity [15], [16].

$$\hat{r}_{ui} = \frac{\sum_{v \in N^k(u)} \text{sim}(u,v) \cdot r_{vi}}{\sum_{v \in N^k(u)} \text{sim}(u,v)} \tag{2.2}$$

Singular Value Decomposition (SVD) [5]. Given a matrix of ratings, **R**, in which rows represent users and in which columns represent items, the SVD algorithm factorizes **R** into several lower-dimensional matrices:

$$\mathbf{R} = \mathbf{USV}^T \tag{2.3}$$

Here, **U** and **V** are orthogonal matrices and S is a singular diagonal matrix, the entries of which are all positive and placed in descending order of magnitude [17]. Some of these entries are small and can be ignored. If the corresponding columns of **U** and **V** are also removed, the result is a matrix that represents a reduced version of the original matrix **R**. If the original matrix is reduced from rank r to rank n, where $n < r$, we can represent the resulting matrix as:

$$\mathbf{R}_n = \mathbf{U}_n \mathbf{S}_n \mathbf{V}_n^T \tag{2.4}$$

Matrix $\mathbf{R}_n$ is now the closest rank n approximation to the original matrix $\mathbf{R}$ [18]. The property can be used to predict unknown ratings by imputing the missing values in the original matrix (e.g. by the mean of the known ratings). After performing an SVD on $\mathbf{R}$, the approximated values for the unknown ratings can be used as a prediction. While this method tends to perform well, it should be noted that the data imputation step introduces noise in the data because the imputed values are treated as actual ratings during factorization. Therefore, it has a negative impact on the overall predicted ratings [17].

To overcome this challenge, regularized SVD (RSVD) was introduced [19] [17]. RSVD only uses known rating values to generate predictions, which removes the need of imputing ratings. RSVD factorizes a rating matrix into two latent feature matrices, a user matrix and an item matrix. Based on this, a rating can be predicted according to equation 2.5:

$$r'_{ui} = \mathbf{p}_u^T \mathbf{q}_i \tag{2.5}$$

Here, $\mathbf{p_u}$ and $\mathbf{q_i}$ represent the latent feature vectors of user u and item i respectively. r'_{ui} is the rating that is predicted for the interaction between user u and item i. The latent matrices of the users and items are learned by minimizing the loss of the generated predictions. L2 regularization is used as a regularizer for RSVD [20]. A form of gradient descent is generally used to find the parameters that optimize the model.

A shortcoming of RSVD is the fact that the method does not take into account structural biases that occur in the data. This is the case for rating data, as some users give structurally lower ratings than others. A method that overcomes this limitation is Improved Regularized SVD (IRSVD). IRSVD introduces two bias parameters to RSVD, of the form 2.6:

$$r'_u i = b_u + b_i + \mathbf{p}_u^T \mathbf{q}_i \tag{2.6}$$

Here b_u represents a bias term for user u and b_i represents a bias term for item i. The parameter learning process is similar to that of RSVD. In the thesis, we benchmark the IRSVD method, because it has consistently shown a better performance in recommender sytems. For simplicity, we will refer to it as *SVD*.

SVD++[21]. Singular Value Decomposition only takes into account the values of known ratings when predicting unknown ratings. However, the sole observation that a user rated an item (regardless of the value of the rating) may also provide useful information. After all, the item was "valuable enough" for the user to interact with. To take this into account, SVD++ extends IRSVD by including implicit ratings in the form:

$$r'_{ui} = \mu + b_u + b_i + \mathbf{q}_i^T \left(\mathbf{p}_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \tag{2.7}$$

In this equation, b_u and b_i again represent the user and item biases. μ is a global bias. $N(u)$ is the set of items that user u rated. y_j are the latent factors of the items in $N(u)$.

The parameter learning strategy that corresponds to this method is again similar to that of RSVD.

Co-clustering [22]. This is another collaborative filtering method based on clustering. Using clustering methods, such as k-mean clustering, users and items are assigned to clusters (C_u and C_i) and co-clusters (C_{ui}). Using these, we can generate predictions as:

$$\hat{r}_{ui} = \overline{C_{ui}} + \left(\mu_u - \overline{C_u}\right) + \left(\mu_i - \overline{C_i}\right) \tag{2.8}$$

Here, $\overline{C_{ui}}$ is the average rating of co-cluster C_{ui}, and $\overline{C_u}$ and $\overline{C_i}$ are the average ratings of clusters C_u and C_i respectively.

SLIM [23]. This is a linear optimization method for generating recommendations. It estimates unknown ratings of items as a sparse aggregation of the ratings for items that are known. For a user u and item i, and a user-item rating matrix $\mathbf{R}$ from which elements are described by r_{ui} this is defined as:

$$\hat{r}_{ui} = \mathbf{r}_u^\top \mathbf{w}_i \tag{2.9}$$

Here $r_{ui} = 0$ and $\mathbf{w}_i$ is a sparse aggregation coefficient vector. Using matrices, this model can be expressed as:

$$\hat{\mathbf{R}} = \mathbf{RW} \tag{2.10}$$

To estimate the matrix $\mathbf{W}$, the following problem should be optimized:

$$\begin{array}{ll} \underset{W}{\text{minimize}} & \frac{1}{2}\|\mathbf{R} - \mathbf{RW}\|_F^2 + \frac{\beta}{2}\|\mathbf{W}\|_F^2 + \lambda\|\mathbf{W}\|_1 \\ \text{subject to} & \mathbf{W} \geq 0 \\ & \text{diag}(\mathbf{W}) = 0 \end{array} \tag{2.11}$$

Where $\|\mathbf{W}\|_1$ is the ℓ_1-norm of $\mathbf{W}$, and $\|\cdot\|_F$ is the matrix Frobenius norm. λ and β are optimization parameters. The usage of sparse aggregation causes the method to be efficient.

2.2. Evaluating Recommender Systems

Evaluating recommender systems is important to quantify their quality. Evaluation can be done with respect to different goals. A goal that is used for determining a recommender systems performance is its accuracy. Metrics that measure accuracy provide a standard and efficiently computable overview of the performance. Rating-based and ranking-based recommender systems both have their own accuracy measures. Beyond accuracy, there exist other metrics as well, such as novelty and diversity, which quantify the characteristics of recommender systems with respect to other goals.

2.2.1. Rating-based accuracy metrics

Rating-based accuracy metrics operate directly on the rating values to measure the accuracy of a model. One such metric is the mean squared error (MSE). Let r_{uj} be a

ground truth rating provided by a user to an item, and $\hat{r}_{uj}$ the estimated rating, then $e_{uj} = \hat{r}_{uj} - r_{uj}$ is the error for this prediction. The MSE is defined as:.

$$MSE = \frac{\sum_{(u,j)\in E} e_{uj}^2}{|E|} \tag{2.12}$$

where E is the set of all user-item interactions. Often, the root mean squared error (RMSE) is used, which can be found by taking the root of the MSE.

Squaring the error causes negative values to become positive, such that positive and negative errors cannot cancel each other out. However, this operation also causes the metric to be sensitive to large errors. This property is not always desirable, for example in a situation where there exist a small number of extreme outliers. For such cases, the mean absolute error can be used:

$$MAE = \frac{\sum_{(u,j)\in E} |e_{uj}|}{|E|} \tag{2.13}$$

2.2.2. RANKING-BASED ACCURACY METRICS

For ranking-based recommender systems, it would be inaccurate to determine their performance based on large unordered lists of ratings. Therefore, ranking-based metrics evaluate ordered lists of (top) N number of recommendations, based on both the rating of individual recommendations and their position in the list. Ranking-based metrics can be divided into correlation-based metrics, utility-based metrics and ROC-based metrics. We focus on utility-based metrics, as these relate most to the work in this thesis. Details on the other metrics can be found in [24].

UTILITY-BASED METRICS

Utility-based metrics combine ground-truth ratings with predicted rankings to determine the accuracy of recommendation rankings. They rely on the assumption that every item has a certain utility to the user. Items with higher ratings have higher utilities, as well as items with high ranks. The optimal utility for the user is naturally provided by a ranking, in which the items with the highest ratings are ranked the highest.

R-score. Concretely, the utility of a ranked list of recommendations can be calculated according to Equation 2.14 for a user u and an item i [24]:

$$F(u,i) = \frac{\max\{r_{ui} - C_u, 0\}}{2^{(v_i-1)/\alpha}} \tag{2.14}$$

In this formula, r_{ui} represents the ground-truth rating that user u assigned to item i. C_u represents the break-even rating value for user u. This value is often set to the mean of the ratings provided by u. Combined, the expression $\max\{r_{uj} - C_u, 0\}$ represents the rating-based utility, which increases with growing rating values. The ranking-based utility is defined by the expression $2^{(v_i-1)/\alpha}$, where v_i represents the rank of item i. α is a so called half-life parameter, which specifies the rank that has a probability of 0.5 of being viewed by the user [25]. In contexts where users generally view many recommendations

before deciding (e.g. car buying), this parameter becomes larger. A higher ranking causes the equation $2^{(v_i-1)/\alpha}$ to grow, such that the overall utility becomes smaller.

An important utility-based evaluation metric for recommender systems is the $R-score$, which can, for a particular recommendation list of length L and for a particular user, be calculated by summing the utilities of the items in the list:

$$\text{R}-\text{score}(u) = \sum_{i \in I_u, v_i \leq L} F(u,i) \tag{2.15}$$

The overall $R-score$ for a list of recommendations for different users can be found by summing the individual $R-scores$ of the different users.

NDCG. The downside of using the $R-score$ as a measure to evaluate recommender systems is the exponential decrease in utility when the rank of items decreases. This causes lower items to be scored with small utility values. This, however, is only valid under the assumption that users' interests are generally limited to the higher-ranking items. While we can assume this in some practical scenarios, it might be not be the case for others. For such scenario's, other metrics might be more suitable. One of such metrics in the discounted cumulative gain (DCG), which is defined as follows for a user u, a set of items I_u and a recommendation list of length L (Eq. 2.16):

$$DCG = \frac{1}{m} \sum_{u=1}^{m} \sum_{j \in I_u, v_j \leq L} \frac{g_{uj}}{\log_2(v_j + 1)} \tag{2.16}$$

In this equation, v_j is defined as the rank of item j in set I_u. g_{uj} represents the utility of item j to user u and is defined in Equation 2.17 [24].

$$g_{uj} = 2^{rel_{uj}} - 1 \tag{2.17}$$

Here, rel_{uj} is the ground-truth rating for user u and item j.. As the DCG value depends on the chosen size of the recommendation list and on the scale of ratings, the DCG can be normalized to allow for easier comparisons. This can be done by dividing the DCG by the DCG of the ideal ranking (IDCG), which can be found by calculating the DCG of the list of recommendations ordered by the ground-truth rankings. This results in the normalized discounted cumulative gain (NDCG):

$$NDCG = \frac{DCG}{IDCG} \tag{2.18}$$

2.2.3. BEYOND-ACCURACY METRICS

The accuracy of a recommender system provides insights into the similarity between predictions and ground-truth ratings. However, this does not guarantee the quality of recommender systems, because there are many more components that can cause recommendations to be good or effective. Beyond-accuracy metrics measure the effectiveness of recommender systems with regard to other objectives. The beyond-accuracy metrics that we cover are diversity and novelty.

Diversity. This relates to recommendation lists generated by ranking-based methods. If these lists are perfectly optimized for containing the most relevant items, there is a risk that this might result in the recommendations being highly similar to each other. Likely, such a list will not satisfy the user, as it results in very limited choices. We use the diversity metric to measure and optimize the level of variety in a recommendation list. A commonly used method of measuring the diversity in a list is by measuring the average pair-wise distance of the items in the list, L, as is shown in Equation 2.19 [26].

$$\text{Diversity}(L) = \frac{\sum_{i \in L} \sum_{j \in L \setminus \{i\}} \text{dist}(i, j)}{|L|(|L| - 1)} \tag{2.19}$$

Here, L represents a list of recommendations. Concretely, the term in the numerator sums the pairwise distances between all items in L. The term in the denominator divides the total of the pairwise distances by the number of pairs in L, consequently obtaining the average pairwise distance. We observe that this method does not take the rank of items into account. This could pose a limitation for situations where users only consider a small amount of the highest ranked recommendations in a list. In such applications, it could be possible that the diversity in high-ranked items is more important than that in lower-ranked items. Different distance measures can be used, depending on the available information. If the items have explicit content features, the Jaccard distance [27] is an appropriate measure. If such features are unavailable, it is possible to measure distance between items using their rating vectors. For this, the Hamming distance or the complement of the cosine similarity could be used [28].

Novelty. This represents how unknown or different recommendations are to users. It can be important to include novel items into recommendations, because they can broaden the interests of users. Often, novelty is estimated using the popularity of items. It is then assumed that items that have been rated by a few users, i.e. items that are unpopular, are novel [29]. We can calculate the popularity of an item as:

$$p(i) = \frac{|\{u \in U, r_{ui} \neq \emptyset\}|}{|U|} \tag{2.20}$$

Here, for an item i, the numerator calculates the number of users, u, from the total set of users, U, that rated i (i.e. the number of users for which $r_{ui} \neq \emptyset$ holds). The denominator divides this number by the total amount of users. Thus, the popularity of i is defined as the fraction of users that rated i. We could then define the novelty of an item in different ways, such as by the complement of its popularity: $1 - p(i)$. Another novelty measure for an item is its *self-information* [28], [30], which is defined as the negative log of the popularity: $-\log_2 p(i)$. The benefit of this measure is that it assigns more importance to rarer items, compared to the simpler complement. Using the latter approach, the novelty of a list of recommendations, L can be calculated as:

$$\text{Novelty}(L) = \frac{\sum_{i \in L} -\log_2 p(i)}{|L|} \tag{2.21}$$

where the numerator sums the novelty of the individual items, i, in L. The denominator divides this sum by the total number of items in a rank. In other words, the metric calculates the average novelty of the items in list L.

The assumption that novelty can be estimated using unpopularity indirectly builds on the observation that items with *few* ratings are different from items with *low ratings*. It is important to note however, that this is not necessarily true. Therefore, Equation 2.21 is an efficient way of estimating novelty, but its effectiveness could differ between application domains.

2.3. CONCLUSION

In this chapter, we presented the background material that will be used in the remainder of this thesis. First of all, we provided a general overview of recommender systems, which are systems that learn to predict interesting items to users. Learning this task requires known ratings that users provided to items. These can be either explicit or implicit. We divided recommendation algorithms into rating-based and ranking-based algorithms. While the former predicts numerical ratings that users would provide to items, the latter produces ranked lists of top-N recommendations. We covered several different rating-based algorithms, global average recommendation, item-KNN, user-KNN, SVD, SVD++, co-clustering and SLIM. Ranking-based versions of these algorithms predict item ratings and rank the items according to those. Next, we discussed metrics for evaluating their performance. We presented different accuracy metrics, namely the MSE and MAE for rating-based systems and the R-score and NDCG for ranking-based systems. Moreover, we discussed several beyond-accuracy metrics that evaluate recommender systems using objectives other than accuracy. To this end, we presented ways to measure diversity and novelty of recommendations.

3

Literature Review

In the previous chapter, we presented the background knowledge used in this thesis. In this chapter, we provide an overview of the available literature related to this thesis. This knowledge will be used to understand the context of this work. Section 3.1 explains how machine learning has been used to mitigate climate change. Section 3.2 covers the existing works on multi-objective recommender systems. Section 3.3 explains how recommender systems have been used as a *nudging* tool. Section 3.4 concludes this chapter.

3.1. Machine Learning for Climate Change

Throughout recent years, substantial research has been done on using machine learning against climate change. The work in [31] divides the research in three categories: mitigation, adaptation and tools of action. Mitigation relates to reducing emissions and adaption concerns preparations for coping with the consequences of climate change. Personal tools can assist in decision making and action taking. We focus on mitigation, as that relates the most to this thesis.

3.1.1. Mitigation.

Mitigation has been researched in a variety of fields. Therefore, the studies concerning mitigation can best be grouped by their application domains.

Electricity systems. These are responsible for about 25% of human emissions [32]. Several previous studies enable the use of low-carbon electricity sources (e.g. solar panels, wind turbines), by forecasting their supply [33]–[38], by forecasting and scheduling the demand [39]–[42] and by researching how machine learning can be used to develop materials that store energy [43], [44]. Other studies focus on reducing the negative effects of current electricity systems. However, [31] rightfully notes that green electricity initiatives can hardly be applied to areas where electricity systems are of low quality and where data on these systems is scarce. Therefore, previous work researched also how to use machine learning to improve energy access [45] and how to apply machine learning

in situations where electrical data is scarce [46].

Transportation. Systems of transportation are complex and account for approximately 25% of global emissions [47]. Figure 3.1 shows a variety of ways of applying machine learning to reduce GHG emissions resulting from transportation, as identified by [31].

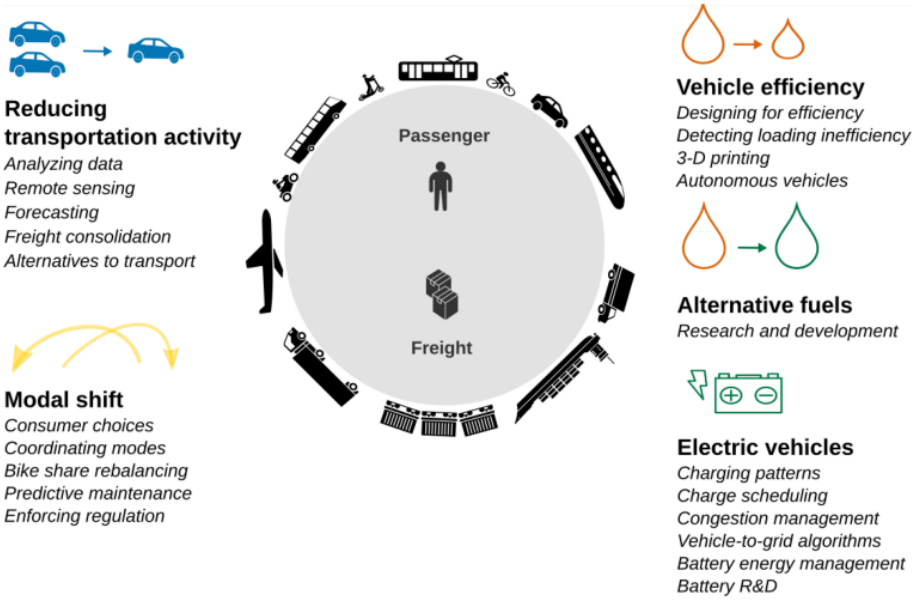


Figure 3.1: Methods of utilizing machine learning to reduce emissions in the transportation sector. Image from [31].

A variety of methods is devoted to reducing transportation activity. Examples are methods for identifying and understanding traffic patterns [29], [48], ways of forecasting traffic [49] and solutions for optimizing the usage of shared mobility [50], something for which recommender systems also have been used [51]. For the latter, however, it is uncertain whether it will actually reduce emissions, as it could also stimulate travellers to choose modes of private transport (e.g. cars, scooters) over public transport [52]. Machine learning also has a significant role in the development of efficient and electric vehicles. Examples are methods for efficient engine design [53], autonomous vehicle development [54] and forecasting electric vehicle battery states [55]–[57].

Solutions for shifting transportation modes have been researched as well and can be divided into two categories: understanding user preferences and stimulating the use of low-emission options. Both naturally relate closely to recommender systems, as these too utilize user preferences to support and stimulate decision-making. To model user-preferences in transport, (semi-)supervised learning techniques have been researched in offline settings [58]–[60]. The downside of such offline approaches is that they require users to provide preference information at a single period in time, which can be a signif-

icant burden. To mitigate this challenge, the work in [61] provides an online approach to modelling transport choices. Other works go even further, by providing entirely automated solutions, based on GPS [61]–[63] and social media data [64]. Works that have investigated methods of stimulating low-emission options, such as public transport, present a wide variety of different approaches as well. The work in [65] aims at improving low-emission solutions, by applying machine learning to assist in maintenance. Other works improve the use of shared mobility [66], [67] and forecast travelling and arrival times of different transport modes to improve their usability [68]–[70].

All these works relate to improving the quality and ease-of-use of low-emission solutions, with the goal of stimulating more sustainable decision-making. Even though the two are related, this approach is somewhat different from the more direct approach recommender systems take. The work in [71] proposes a recommender system that nudges commuters to make more eco-friendly decisions. This is done by assigning a utility value to every travel option based on it's total duration, the duration that a traveller has to bike or walk, the comfort of a trip and the estimated CO_2 emissions generated by taking the trip. A qualitative evaluation of showed that users were positive about the solution, it's accuracy, and eco-friendliness. Also, users indicated that they used the solution often, which shows that green recommender system have the potential to be used in practice. The work differs from this thesis in terms of both the research focus the and methodology. Its focus is on researching the practical usability of green travel recommendations through a user study. To achieve this, the authors develop a software application and generate recommendations using a set of domain-specific rules. We however, more fundamentally research the integration of greenness into recommendation pipelines. To this end, we develop a dataset and evaluation metric, we benchmark existing algorithms and propose a method of integrating greenness into existing recommender systems.

Buildings and Cities. The development and maintenance of buildings and cities is responsible for substantial emissions [31]. Research has been focusing on the reduction of such emissions using machine learning. One area of application is the design of new buildings and the improvement of existing ones. Here, works have explored methods of forecasting energy demand of buildings [72], [73] and appliances [74], [75]. These can be used improve the energy use. Others have investigated solutions to reduce emissions by making buildings and cities smart. Examples of this are studies that use machine learning to predict whether rooms are occupied, such that heating and lighting systems can be used efficiently [76], [77]. Also, studies have investigated how machine learning can reduce emissions on the level of urban planning. Models have been tested to predict energy demands for entire cities and neighbourhoods [78], [79]. These models make it possible to develop systems such as district heating and cooling, which are less-emitting, centralized units that can supply entire districts and neighbourhoods with heating and cooling [31].

It is important to note that the majority of the presented techniques are (implicitly) designed for the global north of the world. They namely all use data and infrastructure sources that are less available in the global south. The most important challenges related to urban emissions, however, are expected to arise from this southern part [31]. Therefore, it will be important that techniques of generalizing solutions are developed.

Industry. Industrial processes and logistics are responsible for tenacious emissions. At the same time, they are generally closely monitored, resulting in much available data, allowing for the use of machine learning to reduce emissions [31]. Machine learning is applied to reduce the energy use of industrial factories. Some studies have developed methods for improving the efficiency of control systems, using techniques such as image recognition and neural networks [80], [81]. Others use predictive maintenance to reduce the amount of damaged and unusable produced goods. There, also, exists works that focus on reducing energy required for industry [82], [83]. Overproduction is another important cause of industry-related climate emissions. With demand prediction, however, such overproduction can be mitigated, as has been shown in several works [84], [85]. Lastly, the work in [31] indicates that also in this domain, recommender systems can have a large impact. The possibilities lie mostly in directing customer towards green purchases. According to the authors, the availability of data is the bottleneck. They state that to make recommender systems useful, it is required that publicly available datasets contain the emissions of products. This thesis provides a first solution for this, as one of the goals is creating such a dataset.

3.2. MULTI-OBJECTIVE RECOMMENDER SYSTEMS

Multi-objective recommender systems are systems that can generate recommendations by simultaneously optimizing several different objectives [86]. In this section, we discuss different techniques and applications of multi-objective recommender systems. As multi-objective recommender systems are closely related to multi-objective optimization, we start off by a discussion about this topic.

3.2.1. MULTI-OBJECTIVE OPTIMIZATION

Many problems require balancing between multiple objectives. When buying a car, for example, one might want to find an option that is fast, cheap and good-looking. While there is likely no car that completely fulfills all three requirements, it is important to find an option that optimally balances them. This can be seen as finding non-dominated, or Pareto optimal, solutions. That is a task for multi-objective optimization (MOO), which is described as the process of simultaneously optimizing a collection of objective functions [87]. While a wide variety of MOO techniques are available, they can be divided into scalarization-based and population-based heuristic methods[86].

Scalarization-Based Methods. These methods transform multi-objective problems into single-objective problems. This is useful, as it allows for applying classical (single-objective) optimization techniques. An often-used strategy for this is the weighted-sum method [86], [88], [89], which assigns weights to the different objectives. The weights represent the importance of the corresponding objective. After assigning weights, the different weight-objective combinations can be merged into a single objective using a weighing technique such as the weighted sum. For a specific set of weights, this results in a Pareto optimal solution. A different commonly used scalarization technique is the ϵ-constraint technique [90], [91]. This technique optimizes only over a single objective function,

while treating the others as constraints. Other notable scalarization methods are goal programming [92] and the Tchebycheff approach [93].

While scalarization techniques are straightforward and efficient, they also have disadvantages. One of these is the selection of parameters, such as the importance weights and the exact ϵ constraints. These should ideally be defined in advance, as searching for the optimal ones is difficult. Even though it is possible that there exists a clear parameter preference, this is not the case for the majority of applications. Applying scalarization techniques results in single Pareto-optimal solutions. If a set of Pareto-optimal solutions is required, scalarization needs to be performed multiple times with different parameters. Lastly, methods based on scalarization may not generate optimal solutions for non-convex problems [86].

Population-Based Heuristic Methods. These use techniques from the domains of evolutionary algorithms [94] and swarm intelligence [95] to find solutions that optimize multiple objectives. There are different ways to include multi-objective optimization into evolutionary and swarm-intelligence based algorithms. The work in [96] introduces a genetic algorithm in which a non-domination criterion is directly included in the fitness function. In other words, non-dominated solutions in populations are regarded as fit. This is a possible way of finding Pareto optimal solutions. A similar strategy is used in [97]. While these methods are more complex and computationally expensive compared to the scalarization techniques, they do not require parameters to be chosen in advance and they can produce sets of Pareto-optimal solutions. This could be useful multiple different optimal solutions are required.

3.2.2. RECOMMENDER SYSTEMS WITH MULTIPLE METRICS

Multi-objective optimization can be used for different recommendation tasks, such as group recommendation [98], [99] and multi-stakeholder recommendation [100]–[102]. We, however, focus on recommender systems that take multiple quality metrics into account, because these relate closely to this thesis. The challenge of such systems is figuratively looking beyond accuracy, to generate recommendations. This is done by optimizing different metrics, essentially balancing them as the optimization of one may negatively influence others. Clearly, this is a well-suited task for multi-objective optimization. Again, we differentiate between works that have used scalarization methods and population-based methods.

Scalarization-Based Methods Most scalarization-based multi-objective recommender systems use the weighted sum method to balance between metrics [103]–[106]. The work in [103] uses a weighted sum approach for reranking recommendation lists to increase diversity. It introduces a single weight parameter that influences the importance of the accuracy and the diversity for the final ranking. A similar strategy is applies by the authors of [106] to generate personalised recommendations based on several different objectives. While being simple and cost-effective, the downside of this approach is again that the weight parameters need to be determined in advance. Often this is done manually. A proper parameter choice differs per application. The authors of [103] visualize this parameter choice by grid searching over the parameter space using a fixed step

size. The authors of [105] approach the problem differently, by pre-selecting a number of parameter choices. This can be more efficient than a full grid-search, but in practice it may require domain knowledge. While automated strategies do exist, they generally require complex and intensive learning algorithms [106], which contradicts one of the largest benefits of scalarization: its simplicity.

Population-Based Heuristic Methods With regard to population-based methods, genetic algorithms are used the most [107]–[111]. By doing this, the authors of [110] and [108] develop a strategy that balances accuracy, diversity and novelty. The work in [109] takes only accuracy and diversity into account. While population-based metrics do not have the challenge of selecting the proper parameters, they do require that a selection can be made from a set of pareto-optimal solutions. After all, a single set of optimal recommendations should be provided. There are several ways of achieving this. The authors of [109] use the hypervolume [112] for this. The work in [111] used a less costly method: the best solutions for individual objective were selected and it was investigated how the selection influenced the objective. The downside of this approach is that it also requires that a manual decision is made, with regard to which objective is more important than others.

3.2.3. MULTIPLE OBJECTIVES IN THIS THESIS

Our work relates to multi-objective recommender systems in that we aim to explore the balance between greenness and accuracy. We, however, choose not to use multi-objective optimization techniques for this. Instead, we use a strategy, similar to that presented in [71] and [113]. As part of our strategy, we generate rating predictions with the goal of optimizing their accuracy. Afterwards, we re-evaluate their utility by calculating a weighted sum between the rating values and the greenness of items. We chose to use this technique, because it is more efficient and provides more control over the experiment than multi-objective optimization strategies. We require this control as we want to understand the trade-off between accuracy and greenness.

3.3. NUDGING WITH RECOMMENDER SYSTEMS

Nudging refers to the process of helping people make better decisions, without forcing them towards particular options [114]. Originally, nudging techniques were only applied to offline settings [115], [116], but more recently, digital nudging techniques have been researched as well [117]. As recommender systems aim to help users make better decisions, they can also be used for nudging. Substantial literature has been published on this topic, researching different nudging techniques in a variety of application domains. The work in [118] identifies several main techniques of nudging users with recommender systems. In this literature review, we cover one of them, the decision structure, as it relates most to the work in this thesis.

Recommender systems can be used to change the structure of decisions and steer the decision-making process. The authors of [71] apply this technique. To direct users towards more sustainable routing options, they rank and select recommendations, such that the shown options are both green and interesting. The work in [113] uses a similar

technique to direct users towards healthier food-choices. Both works strongly relate to the work in this thesis, as they rank items based on their greenness and their predicted value to the user, which is similar to the technique that we propose in Chapter 6. Option ranking is also performed by the authors of [119], which rank wireless networks based on their security. In contrast to the previous works, however, this work generates the ranks using a pre-defined set of rules rules and adds different colours to the recommendations, which present security levels. These are meant as an additional nudge.

3.4. CONCLUSION

In this chapter, we have provided an overview of the literature that relates to the work in this thesis. We explained how machine learning has been used to mitigate climate change. We covered works in the contexts of electricity systems, transportation, buildings, cities and industry. Following this, we described how multi-objective optimization can be used to develop recommender systems with multiple objective metrics. We covered scalarization methods and population-based heuristic methods.

We showed that there are no works that research how greenness can be integrated into existing recommender systems, despite the fact that substantial literature is available on machine learning for climate change. Therefore, the remainder of this thesis will present an entire pipeline for making recommender systems green. To achieve this, we will introduce the first dataset for green recommender systems, we will present several metrics to measure the greenness of recommendations, we will benchmark several algorithms on their greenness and we will present a strategy to improve the greenness of recommender systems.

4

BUILDING A GREEN RECOMMENDER SYSTEM DATASET

In the previous chapters, we discussed background knowledge and related works. In this chapter, we propose and develop a new dataset that can be used to research the footprint of recommender systems. Section 4.1 introduces the challenge of making a green recommendation dataset. Section 4.2 provides the most important definitions of the dataset. Section 4.3 gives an overview of the created dataset. Section 4.4 analyses the created dataset and compares it to other datasets. Secion 4.5 describes how the data was collected. Lastly, Section 4.6 concludes this chapter.

4.1. INTRODUCTION

Datasets are required to develop and assess new recommender systems. Moreover, they can provide insights into user preferences. Movielens [120] and Book-Crossing [121] are two popular examples of datasets that are commonly used in recommender system research to assess and compare algorithms in terms of accuracy, diversity and novelty.

There exists, however, no dataset that contains both user-item preferences, combined with the CO_2 footprint of the items. This is a likely cause for the lack of evaluation of the footprint of recommender systems [31]. Therefore to perform said research, it is required to create such a dataset in the first place; a challenging process that has several facets as discussed in the remainder of this chapter.

First of all, a dataset needs to be created that contains user preferences of items for which their footprint can be estimated. While movies, for example, do have a climate footprint, estimating it would require estimating the footprint of all steps of the movie's production, its allocation in different cinemas and its broadcasting. Naturally, this is nearly impossible, causing movies to be a poor item choice. Second, a challenge is posed by estimating the footprint of individual items in a consistent manner. This can be done in several ways, for example, by using heuristics or by dividing the items into multiple sub-items, and estimating the footprints of the sub-items individually. Which strategy

should be chosen strongly depends on the biggest challenge of the process: finding footprint data. The work in [31] indicates that data on item-footprints is unavailable. Research often focuses on items that hardly relate to wide-scale user preferences, such as raw materials (e.g. coal, oil) and industrial products and processes [122] [123]. Consequently, there is no available information about the footprint of items common to recommender system's research. Lastly, for the practical relevancy of this research, it is important that the domain of the dataset is one where CO_2 data is able to influence user decisions. Coming back to movies, user's decisions are unlikely to be influenced by CO_2, as there often are no similar alternatives to specific movies (e.g. Star Trek is not necessarily a replacement of Star Wars). For fashion, however, customers are often presented with different items that are similar (e.g. t-shirts of different brands with the same colour and fit), allowing them to consider multiple options for their purchase. In this scenario, it is more likely that a CO_2-responsible recommender system would have a more practical use.

By addressing these challenges, we propose the RecipeEmissions dataset, which includes user preferences for food recipes, along with the CO_2 emissions of the recipes. The domain of recipes is chosen because of two reasons:

- Food/recipe recommendation is a well-researched domain of recommender systems [124].
- Recipes can naturally be split into ingredients (with corresponding usage quantities). As the footprints of commonly used ingredients are publicly available, it is possible to estimate the footprints of entire recipes. This can be done by collecting the footprints of the individual ingredients and summing them up according to the respective quantities used in the recipe.

The preference and recipe data will be obtained from the Food.com dataset [9]. This dataset consists of users, recipes, and the ratings that the users have given to the recipes. It also contains the ingredients of the recipes and their usage quantities. The CO_2 data will be obtained from several different sources.

4.2. DEFINITIONS

To account for the carbon emissions of the recipes during recommendation, it is important to characterize how the emissions are made up. Recipes consist of ingredients, the production of which causes CO_2 emissions. We define the carbon emission of a recipe as the sum of emissions caused by producing the ingredients of the recipe. Ingredient emissions are expressed by their CO_2 equivalent (CO_2-eq), which is the equivalent amount of CO_2 (in kg) with the same global warming potential [125], [126]. The ingredient emissions are weighted according to the quantities in which they are used. For example, consider a recipe that includes 300 grams of apples and 10 grams of sugar, with respective emission values of 0.5kg CO_2-eq and 10 kg CO_2-eq, then the total amount of CO_2-eq for this recipe is 0.3*0.5+0.01*10=0.25 kg CO_2-eq.

4.3. FINAL DATASET OVERVIEW

The final dataset consists of 32.093 users, 5.605 recipes and 247.521 interactions. Users and recipes are expressed by their numerical identifiers. The interactions are explicit ratings between 0 and 5 (incl.). The sparsity of the user-recipe rating matrix is 99.88%. The dataset also contains three item features: the name, the ingredients and the CO_2-eq in kilograms. This makes the dataset .the first of its kind.

4.4. ANALYSIS

To get a better feeling for the structure and properties of the dataset, we perform two main analyses: analyses related to dataset properties and to emissions. For the former, we compare the RecipeEmissions dataset with the Movielens100k, Movielens1M [120] and BookCrossing datasets [121]. We also compare the final dataset with the original Food.com dataset.

4.4.1. DATASET PROPERTIES

Statistics. Table 4.1 compares the statistics of the RecipeEmissions dataset with the baselines. The number of users and items, and the number of interactions in the dataset are reduced compared to the original Food.com dataset. This is the result of several operations, applied to reduce the number of recipes and the sparsity of the data (see 4.5). By comparing with the other datasets, we can see that Movielens100K and Movielens1M have significantly less users and items than our dataset, but are more dense. The Book-Crossing dataset is most comparable to the Food.com dataset, having many interactions and being sparse.

The RecipeEmissions dataset has an average rating close to the maximum. This property is inherited from the original dataset and it has been slightly increased by the operations performed on the set. Here, it should be taken into account that the range of the ratings is from 0 to 5. As this is larger than the 1 to 5 range of the Movielens sets, the relative difference between the average ratings is even larger. Such high ratings are likely to influence accuracy metrics such as the RMSE and the NDCG. The RMSE is likely to be lower, because there is less spread in the rating data, making it simpler for algorithms to make more accurate predictions. The NDCG is likely to be higher, because having more high ratings automatically increases the chances that highly-rated items end up high in the ranking. The Book-Crossing dataset stands out with the lowest average rating.

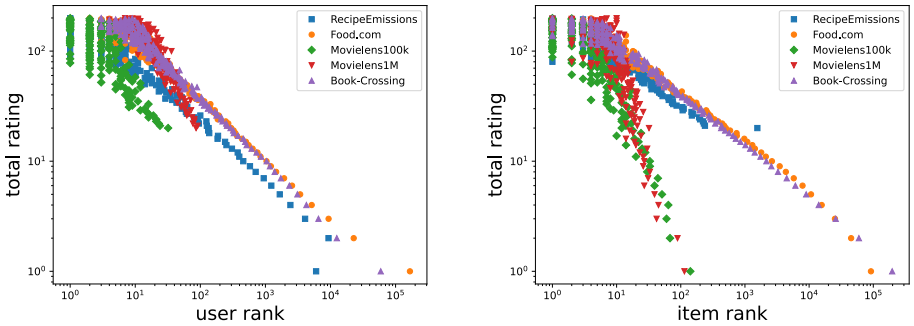
By looking at the median of the number of interactions per item and user, Table 4.1 shows properties in line with the sparsity. The RecipeEmissions, Food.com and Book-Crossing datasets have less interactions per user and per item, compared to Movielens. The RecipeEmissions dataset has a larger median per item, because we limit the number of different recipes (see Sec. 4.5). Overall, we conclude that the statistics of the proposed dataset, and their relative values, are in line with the other often-used datasets.

User and Item Distributions. Figure 4.2 shows the user engagement distribution in Fig. 4.1a (total ratings of each user) and the item popularity distribution in Fig. 4.1b (total rating per item).

Table 4.1: Statistics of the RecipeEmission, Food.com, Movielens100k, Movielens1M and Book-Crossing datasets.

	RecipeEmissions	Food.com	Movielens100k	Movielens1M	Book-Crossing
#users	32090	226570	943	6040	105283
#items	5595	231637	1682	3706	340556
#interactions	247219	1132367	100000	1000209	1149780
sparsity	99.86%	99.998%	93.70%	95.53%	99.997%
minimum rating	0	0	1	1	0
maximum rating	5	5	5	5	10
average rating	4.57	4.41	3.53	3.58	2.87
median interactions per ingredient	27	2	27	124	1
median interactions per user	3	1	65	96	1

From Fig. 4.1a, we observe that all datasets present a long-tail distribution, which implies that the majority of users occur in few interactions. This is common for recommender system datasets. However, the distributions also have differences among them, which corroborate the earlier statistics. Namely, the denser Movielens datasets contain relatively more users with a larger number of ratings, compared to the other sparser datasets. Even though the RecipeEmissions dataset has relatively more users with more ratings compared to the Food.com dataset, the overall distributions remain similar.



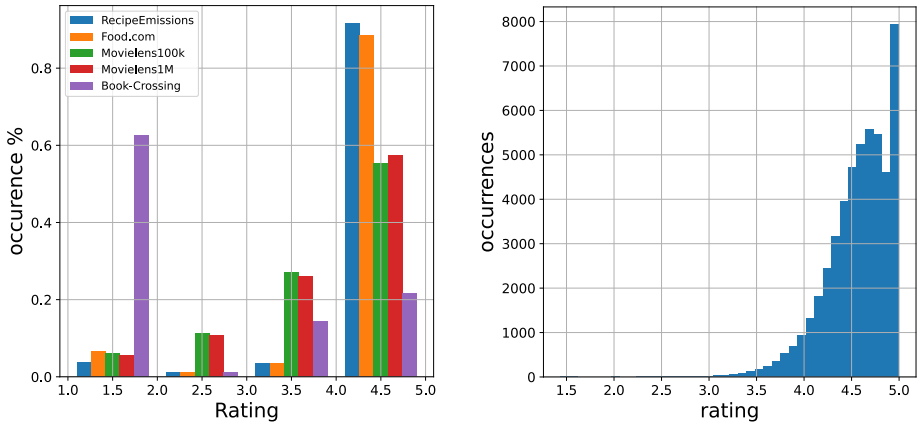
(a) User engagement distribution (total ratings of each user). (b) Item popularity distribution (total ratings of each item).

Figure 4.1: Key properties of the RecipesEmissions, Food.com, Movielens100k, Movielens1M and Book-Crossing datasets. Both figures have log/log scales.

Also 4.1b shows long-tail distributions for all datasets, indicating that the majority of items occurs in small number of interactions as well. Again, we observe, though, that relatively more items from the Movielens datasets have higher numbers of ratings corroborating the density of the sets. Moreover, we observe that the RecipeEmissions dataset does not include items with less than 20 ratings. This is caused by a limit that we impose on the recipes in the RecipeEmissions set; they should occur in at least 20 unique interactions. This decision is further detailed in 4.5.2. Fig. 4.1a shows that the same re-

quirement applies for users in the Movielens datasets. Lastly, both figures show that the distribution of the RecipeEmissions dataset is similar to that of the Food.com dataset, except for a single point in both figures. These points are caused by the removal of interactions, further detailed in Sec. 4.5.2

Rating Distribution. Figure 4.2a shows the rating distribution in the different datasets. As the ranges of the rating values differ in each dataset, we scale all distributions to a range between 1 and 5, which is the smallest existing range (Movielens).



(a) Distribution of ratings in the datasets. The ratings of all datasets are scaled to a range between 1 and 5. Bars between two round numbers, a and b, represent ratings in the range $[a, b)$ (e.g. $[1, 2)$)

(b) Predicted ratings resulting from training a matrix-factorization algorithm on the RecipeEmissions dataset.

Figure 4.2: Rating distributions of RecipesEmissions, Food.com, Movielens100k, Movielens1M and Book-Crossing datasets and rating predictions, generated by the SVD algorithm.

First, we find that the distribution of the two Movielens datasets is similar to a logarithmic increase. This indicates that reviewers are overwhelmingly positive. The Book-Crossing dataset shows an almost the opposite pattern. Reviewers in this dataset appear to be more critical towards items: there is a high percentage of low ratings. Yet a different pattern is shown by the RecipeEmissions and Food.com datasets. There is a significant bias towards the rating of 5. This corresponds to the findings in Table 4.1. The distribution present in the Food.com dataset has remained present in the RecipeEmissions one. A potential challenge that is posed by the bias is that recommender systems learn from the data. This means that it is likely that the predictions of the recommender systems, trained on this data, also contain a bias towards high ratings. To illustrate and evaluate this statement, we have split the RecipeEmissions dataset into a training and a test set, after which we trained a recommender system algorithm on the training set. The algorithm is a simple form of matrix factorization. The distribution of the predictions, generated for the entries in the test set, can be seen in Figure 4.2b. The figure shows a

clear bias towards very high predictions, many of which are even 5. When benchmarking prediction algorithms, especially rating-based ones, this observation should be taken into account.

4.4.2. EMISSIONS

To understand the distribution of the CO_2 equivalents found for the recipes in the dataset, we analyse these as well. Figure 4.3 shows the distribution of carbon footprints of recipes within an interval between 0 and 200. The number of occurrences decreases logarithmically as a function of the increasing footprint. This means that relatively many recipes have relatively small footprints. Figure 4.4 shows a similar image for users: the majority of users have relatively small footprints.

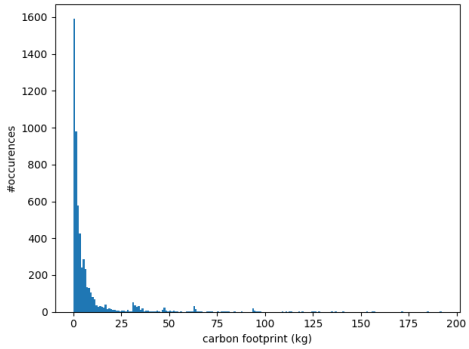


Figure 4.3: Number of occurrences of recipes with different footprints (between 0 and 200 kg)

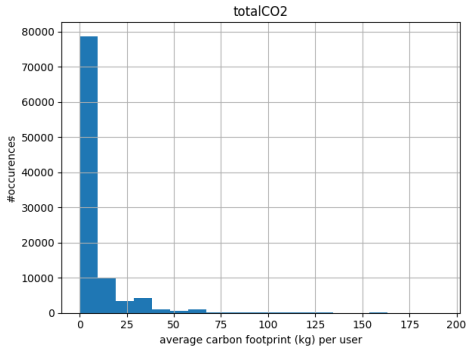


Figure 4.4: Number of occurrences of users with different footprints (between 0 and 200 kg)

Lastly, figure 4.5 shows the footprint distribution of the recipes as a function of the ratings received. There is no apparent difference between the distributions, potentially indicating that the carbon footprint of recipes have no major impact on the ratings assigned to them. This could imply that greener recommendations could be made to users without a negative impact on their accuracy.

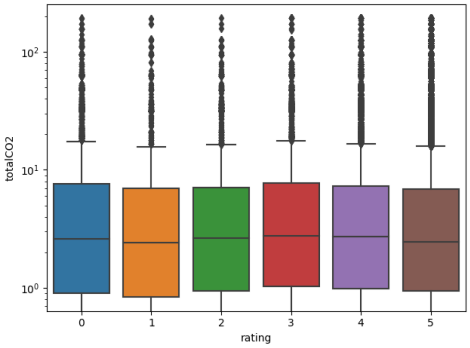


Figure 4.5: Distribution of the footprints per rating (logarithmically scaled)

4.5. DATA COLLECTION

This section describes the creation of the dataset. First of all, the process of collecting ratings is described, followed by the method for obtaining the quantities of ingredients. Lastly, the collection of emissions is covered.

4.5.1. BASE DATASET

A dataset that describes how users rate different recipes is publicly available[1], and was originally created by the work in [9]. The data was collected by scraping the popular recipe-sharing website Food.com[2] and contains 178264 recipes, 25075 users, and 1125284 interactions between them as explicit ratings from 0 to 5 and textual reviews. The sparsity of the dataset is 99.997%. The dataset contains the names and numerical identifiers (id's) of recipes, the identifiers of users, and a list of interactions between users and recipes (user X rated and reviewed recipe Y). The names and identifiers of the ingredients used in recipes are also present in the data. The ingredient quantities are not available and need to be found differently.

4.5.2. PREFILTERING

As a significant number of the users in the dataset have only rated a few recipes, and as a significant number of recipes have been rated by only a small number of users, all recipes rated by less than 20 users are filtered out. Having so many items with few interactions leads to a bias towards a cold start scenario when we benchmark different algorithms. We want to evaluate the greenness of recommender systems in an unbiased way. Therefore, we choose to remove few-occurring recipes. Additionally, finding the emission values of ingredients is an intensive and costly procedure, as we discuss next, therefore, keeping all few-occuring recipes significantly increases the complexity of creating the dataset, without providing much value. Removing few-occuring users and items is something that is commonly done for recommender system datasets. The Movielens datasets, for example, apply a threshold on the number of interactions a user

[1] https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions
[2] food.com

should have to be part of the dataset. In their case, this threshold also has the value of 20 [120]. Besides the few occuring recipes, the recipes for which ingredients are not provided, are also removed. Consequently, all users that did not review the remaining recipes are removed too. This results in 5936 recipes and 99781 users.

Still, there are especially many users that only have 1 rating, as can be seen in figure 4.6. To also prevent a cold-start bias with regard to users, we remove users with only 1 rating without affecting the recipes. To achieve this, for every recipe with more than (the minimum of) 20 ratings, we remove those users that rated only that recipe, while maintaining the threshold of 20 ratings per recipe. This results in a less sparse matrix (99.88 percent). The distribution of the number of ratings of different users (until 100 ratings per user) can be seen in figure 4.7. The overall distribution of the original data (figure 4.6 remains present in the filtered dataset.

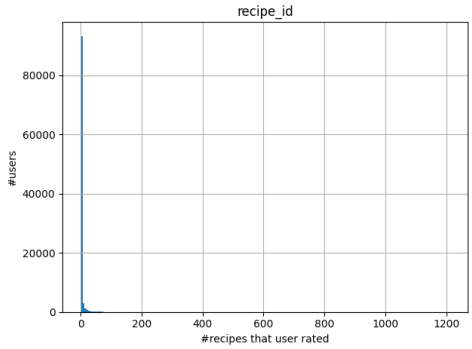


Figure 4.6: Number of occurrences of users having rated different intervals of number of recipes.

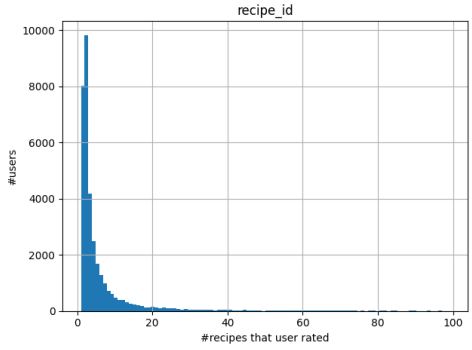


Figure 4.7: Number of occurrences of users having rated different intervals of number of recipes, after removing 1 rating users from recipes with over 20 ratings. Only shows interval 0-100.

We also remove those recipes that are rated by users with a small number of ratings. For every recipe, the average number of ratings of the users that rated the recipe is calculated. Recipes, for which it holds that users have less than 20 ratings on average are

filtered out of the dataset. This forms the final dataset (this is different from our first prefiltering step, where we removed all recipes with less than 20 ratings).

4.5.3. QUANTIFYING INGREDIENTS

To estimate the recipes' emissions, the quantities of corresponding ingredients need to be found. Therefore, we develop a protocol consisting of the following steps:

1. We scrape the Food.com website to find the ingredient quantities, belonging to the original recipe.
2. We remove any inconsistencies in the data.
3. We remove recipes on Food.com containing ingredients that are not present in the dataset.
4. We standardize the measurements of the ingredients provided in different units.
5. Lastly, we manually annotate some ingredients in the data that have no quantity indication.

Scraping Quantities. We scrape the ingredient quantities from Food.com. To achieve this, the urls corresponding to recipes in the dataset need to be generated dynamically. The urls of the website that lead to recipes are made up from a combination of the following elements:

www.food.com/ [tag] / [recipe name] - [recipe id]

The tag can be either *recipe* or *about*, and depends on the recipe. Which tag is used for a recipe differs per case and is information that is not available in the original dataset. We can find the recipe name by decapitalizing the recipe name in the dataset and by replacing all the whitespaces with hyphens (-). Lastly, the recipe id is naturally represented by the id corresponding to a recipe in the dataset. The webpage that corresponds to the URL contains a recipe from which we can retrieve ingredients and quantities. Food.com assigns id's to individual ingredients but they do not correspond to the id's of the ingredients in the dataset. However, the ingredients on Food.com are in the same order as in the dataset. This information aids the scraping process and is used in the following way: we scrape the ingredients on the webpage in the order in which they occur in the dataset. On the webpage, ingredients are described by the used quantity in the recipe, a label containing the measuring unit (e.g., grams, cups) and the name of the ingredient (e.g., water). Therefore, for every combination of recipe and ingredient, we scrape and store the quantity and measuring unit.

Standardizing Quantities. The data on Food.com is not standardized and we should make additional changes before estimating the exact ingredient quantities. First, the scraped quantities do not always provide a single number. Often, such values represent ranges (1-2 cups). In these situations we use the mean of the interval. Second, there are occurrences of quantities that are in a decimal format (2.5 cups) and occurences that

are in fractional format (2 1/2 cups). We convert all such occurrences to decimal formats.

Removing Excess Ingredients. Comparing the scraped data with those available in the dataset shows that in 797 cases the scraped recipes contain more ingredients than in the dataset. The additional ingredients form alternatives or minor additions to the recipes in the dataset, such that the recipes do not change substantially. However, the ingredient quantity scraper relies on the similarity between the ingredient order in the dataset and on Food.com. Therefore, the additional ingredients could lead to high innacuracies for these 797 recipes. While we could have removed the recipes for which the ingredient list in the dataset is smaller than the scraped ingredient list, this would imply removing 12% of the recipes (797 out of 5936). We argue that the negative impact of this on the quality of the data would likely be significant. Therefore, we choose to solve this problem for as many recipes as possible. The strategy that we chose for this is shown in Figure 4.8.

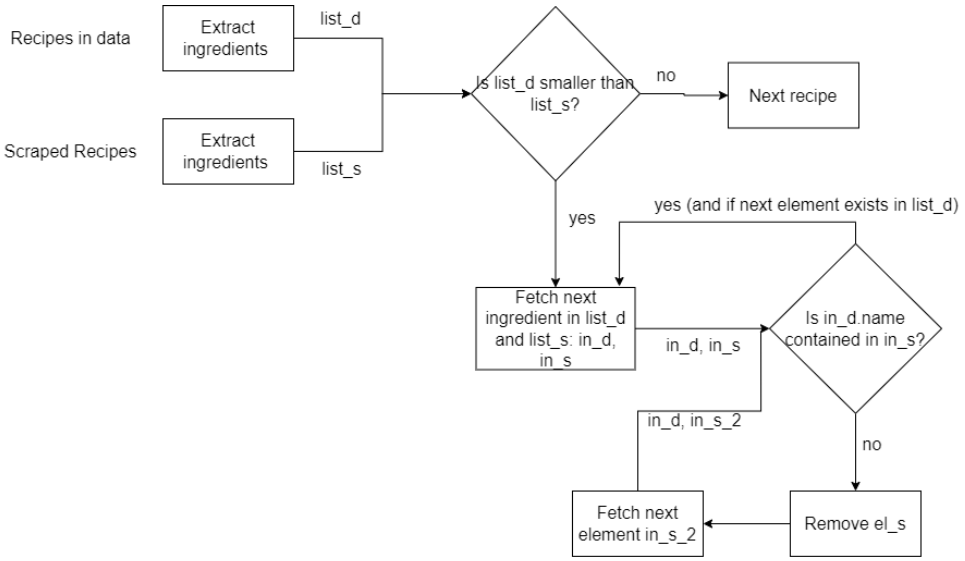


Figure 4.8: Linking the scraped ingredients to their proper counterparts in the data.

The ingredient lists of the scraped recipes and the original recipes are compared. There are no occurrences of original recipes with more ingredients than the scraped variant. Therefore, for every recipe, we perform the following steps:

1. We check if there are more ingredients in the scraped recipe than in the dataset.
    (a) If so, the recipe lists are compared with each other. For every pair of ingredients, it is evaluated if the name of the ingredient from the original dataset is contained in the name of the scraped ingredient. This is possible, because the ingredient names in the dataset often match, or are a subset of the names of the scraped ingredients.
        i. If this condition is true, we consider the scraped ingredient to correspond to the ingredient in the data.

Table 4.2: Conversion rates of several units to grams. The values are retrieved from [127]

unit	conversion rate to grams
1 ounce	28.35
1 pound	453.59
1 Pint	308.38
1 Pinch	0.36
1 Cup	171.16
1 Teaspoon	3.91
1 Tablespoon	11.74

ii. If this condition is not true and the names are different, it can be assumed that the scraped ingredient is not present in the ingredient list of the original dataset. We remove the ingredient from the list.

(b) If there are not more ingredients in the scraped recipe than in the dataset it means that the items are in the correct order and that we do not have to change anything.

After this process is completed, all ingredients and respective quantities in the original dataset are linked to their scraped counterpart.

Unit Conversion. The scraped data contains the ingredient quantities and their corresponding units (ounce, lb, etc.). The quantities are converted into grams, because the CO_2 equivalent of ingredients is expressed per kilogram of ingredient [126] and because the ingredients are often used in small quantities (small number of grams). For the unit measures ounce and pound this is straight forward, as standard conversion rates exist. These can be seen in table 4.2. The other conversion rates in the table have no standard conversion rates to grams. The conversion rates that are applicable to these unit measures differ per ingredient, and depend on their densities. Finding the density values for all the required ingredients in the dataset is challenging. Therefore, it is chosen to estimate generally applicable conversion rates for each of these items, based on the densities of ingredient categories that are often measured using the specific unit measure.

For example: we observe that the teaspoon measure is often used to indicate quantities of liquids, sugars, flowers, oils and spices. Therefore, the densities of several often used products within these categories are used to calculate the weight of a teaspoon (in grams). Finally, the mean of the outcomes for the different products is taken as a general conversion rate. For a teaspoon, the used ingredients are, in order respective to the named categories, water (4.93 grams), raw sugar (4.74 grams), all-purpose flower (2.61 grams), olive oil (4.53 grams) and cinnamon (2.76 grams). This results in a total conversion rate of 3.91 for a teaspoon to grams. This process is repeated for all of the different unit measures. The categories, representative ingredients and weights that are used for this, can be found in Appendix A.

Finding Quantities Manually. Lastly, there exist ingredients for which no quantity is given. This is often the case for products consisting of portions or pieces (e.g. 2 chicken breasts, or 1 bell-pepper). The weights of ingredients falling into this category are found manually: for every product, an annotator is asked to estimate the weight of a portion of the product by searching for it on the internet. The first search results are used for this. This step is done for all the 4010 ingredient usages for which no unit indication is available. There are more unique recipe usages than unique recipes, because the quantity indication can cause single ingredients to occur multiple times (e.g. a bag of celery and a bunch of celery involve the same ingredient, but both are different usages). Next to the portion weights, the annotators provide their source and an estimation of their certainty (between 1 and 10) that an annotation is correct[3]. Annotations with certainties below 5 are manually inspected afterwards, resulting in 73 corrections.

4.5.4. QUANTIFYING EMISSIONS

For every ingredient in the recipe, we collect the corresponding CO_2 equivalent. Because of the large number of ingredients in the dataset (2853), the process is divided into three steps. First, a preprocessing step removes insignificant ingredients. Second, an automated search finds the CO_2 of the ingredients. Third, a manual search is done for the emission rates of ingredients for which no emission data could be found automatically.

Insignificant ingredients. To remove insignificant ingredients, we do a search through all the recipes. For every ingredient, we determine its maximum usage quantity over all recipes. If there are no recipes that use more than 50 grams of the ingredient, the ingredient is deemed insignificant and we remove it. The threshold of 50 grams filters ingredients that do not have a large CO_2 equivalent, such as spices and flavourings [128], [129]. Therefore, the threshold has no significant impact on the overall CO_2 values of the recipes. In total, 705 ingredients are removed.

Automated search. For the automated search, we use the website healabel.com[4]. This website contains the CO_2 equivalent for a large number of ingredients. We use this website, because it is the only available source that has information on the CO_2 equivalents of a wide variety of different ingredients. On the website ingredients have a dedicated page, with a dedicated URL. To scrape the emission data, we can use the structure:

https://healabel.com/ + [first letter ingredient] + -ingredients + / + [ingredient name]

If an ingredient name consists of multiple words, these are separated by a hyphen (-). There exist several exceptions to this default scheme. For example, for some ingredients the ingredient names are appended with the letter 's', such they present the plural version (e.g. nut becomes nuts). There is no clear pattern with regard to when this is the case. Therefore, we first try the version of the ingredient without the 's'. If that does not result in a valid page, we try the version with the 's' appended.

Moreover, a large number of the ingredients are variations of others. Many of such

[3] The estimated quantities, sources and certainties are provided in the RecipeEmissions dataset.
[4] healabel.com

variations add an adjective to the original ingredient name, such that it describes a specific instance. For example, Pepper occurs as pepper, red pepper, green pepper, orange pepper, etc. Comparing the CO_2 equivalents of these ingredients shows that they do not differ significantly. Therefore, to reduce annotation costs, we choose to gather all such different variations under the ingredient's base name (e.g. pepper). This step can only be performed for situations in which an adjective precedes an ingredient name. There are also scenario's in which the ingredient name represents the adjective. Blindly using such an adjective to represent the full ingredient name is often inaccurate. For example, in the case of meat substitute, using the adjective (meat) would be hardly representative for the CO_2 emissions generated by producing the ingredient (meat substitute). So, we do not to apply the technique to such situations. Lastly, empirical analysis of ingredients with more than 2 words showed that the last word represents the ingredient. These observations result in the diagram in figure 4.9, which describes the generation of ingredient names for the URLs in the scraping process. Ingredients that cannot be found automatically are marked as 'not found'.

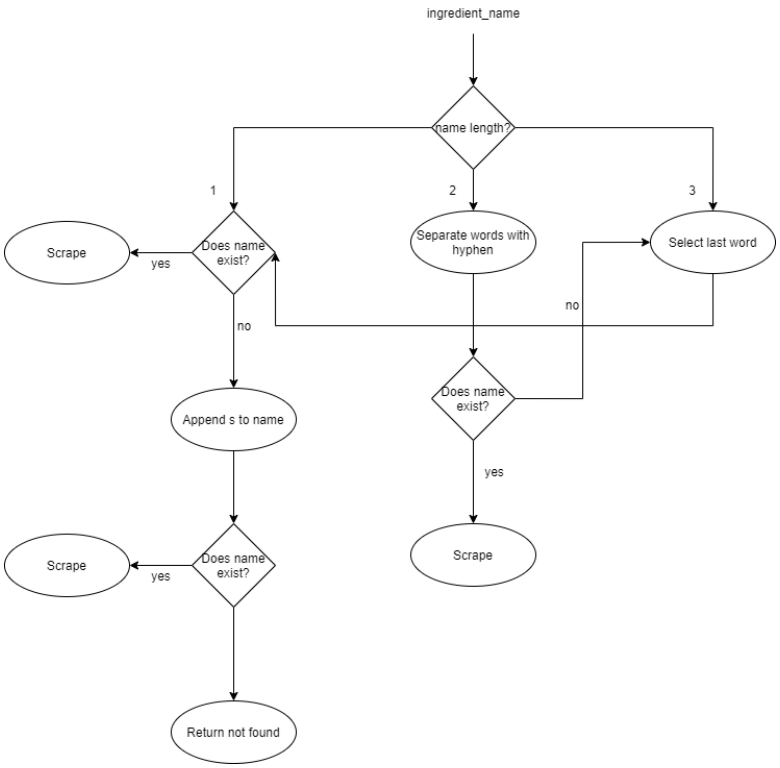


Figure 4.9: The process of generating ingredient names for URLss, used for the scraping process

Manual search. As many as possible of the remaining ingredients are found manually. These ingredients are first searched on Healabel.com to standardize the data as much as possible. If an ingredient is unavailable on this website, it is manually looked

for in published literature. This process is performed by entering the following query into Google Scholar[5]:

[recipe name] CO_2 footprint.

The resulting papers are checked first and then ordered by the number of citations and the first paper that contains the footprint information is used to retrieve the data [130]–[138].

To evaluate the quality of the dataset, 100 ingredients were randomly selected. These ingredients were manually inspected to determine the accuracy of their CO_2 equivalents. The accuracy was judged by evaluating the similarity between the name of the ingredient in the dataset, and the corresponding name of the ingredient for which the CO_2 was retrieved. This resulted in a 92% accuracy.

4.6. DISCUSSION

We proposed the first recommender system dataset containing emission values for the items in CO_2 equivalents. This dataset that can be used for researching green recommender systems. It contains 32.090 users, 5.595 recipes and 247.219 interactions. The dataset is intended as a standard solution for evaluating the greenness of recommender systems. It could prove to be highly valuable for researching this aspect. In the remainder of this section we discuss the choices we made. It should be noted that this dataset acts as a baseline, and could be improved in further research.

4.6.1. ANALYSIS OF DATASET

We analyzed the RecipeEmissions dataset and compared it to several other commonly used datasets, showing that their distributions are similar. However, we also discovered that there exists a bias towards high ratings in the dataset. Using a simple experiment, we found out that the predictions of a recommender systems also contain this bias. This will likely influence our benchmarking experiment. The bias namely results in less spread in the prediction data, which makes it easier for recommender systems to generate predictions that are accurate for many interactions. Moreover, we employed several techniques to reduce the sparsity of the dataset to prevent a bias towards a cold-start scenario. However, there still remain many users that have rated only a small number of items. In the benchmarking experiment this could have a negative effect on the accuracy of algorithms that make use of user similarity, as it can be difficult to determine the similarity users with few ratings.

4.6.2. ANALYSIS OF CREATION

The dataset creation process knows several limitations. First of all, the ingredient names used to retrieve CO_2 data were standardized in a number of cases e.g., adjectives were removed. While this improves the feasibility of finding CO_2 equivalent for many ingredients, this choice also automatically reduces the quality of the dataset, as the original ingredients are more specific. Currently, however, it is impossible to obtain CO_2-eqs of all

[5] https://scholar.google.com/

recipes on a specific level, as there is a lack of data in this regard. Second, the CO_2 rates for the different ingredients were collected from a variety of sources, which affects the quality of the dataset. The CO_2 equivalent of a product, for example, can also include the CO_2 emissions generated by shipping the product. As different data sources have been used, some of which originate from different parts of the world, the actual CO_2 emissions differ. Third, many of the emission values are retrieved from Healabel.com. The reason for this is that this website is the only platform with CO_2 equivalents for a wide variety of ingredients. The website, however, is not scientifically backed. This naturally increases the risk that emission values from the website are inaccurate. We mitigated this partially by taking samples from the website to compare them. This showed that the relative differences between the CO_2 equivalents of the ingredients in the samples were as expected, which is the most important part for recommender system research.

Lastly, the quantities of a number of ingredients were found by annotators. This automatically introduces the risk of inaccuracies in the data, as different annotators can have different approaches to data collections. We tried to mitigate this as much as possible by providing them with detailed instructions. However, it is still possible that inaccuracies have resulted from the annotation process. Naturally, this could alter the total CO_2-eq of recipes.

4.6.3. FUTURE WORK

As part of a future research, it would be valuable to research standardized techniques for creating other datasets to research green recommender systems. It is important that datasets for different applications and domains are created. In order to make green recommender systems practically applicable, we argue that it is vital that the creation of datasets can be done with minimal costs. Instead of finding the exact CO_2-eqs of items, it would, for example, be possible to indicate the greenness of items with a score, similar to ratings. This would ease the process of estimating footprints. In Chapter 5, we show that our greenness metrics are compatible with this. However, it still remains relevant to determine the exact footprint of products. This is currently difficult. Therefore, it is important that research in this direction is done as well: either by researching new and simpler methods for quantifying CO_2 emissions or by researching the CO_2 equivalent of individual goods.

5

Benchmarking Recommender Systems

This chapter contains one of the core contributions of this thesis. We benchmark different commonly used recommender systems in terms of emission rates and accuracy. This chapter is structured as follows. Section 5.1 explains the algorithms that we benchmark. Section 5.2 explains the metrics created for measuring the greenness of recommender systems. In Section 5.3, we then describe the setup of the experiment. The results of this experiment are provided in Section 5.4. In Section 5.5 we discuss the experiment, results, and their implications.

5.1. Algorithms

There is a variety of rating and ranking-based recommender system algorithms. Evaluating the CO2 emissions of all algorithms would be infeasible. However, many algorithms can be clustered into categories. Therefore, several commonly used algorithms are selected from several different categories. For generating predictions, we use the following algorithms. The algorithms are explained in more detail in Chapter 2:

- **Global average**: we calculate the average rating that users provided to items and use it as the predicted rating for all unknown interactions. We use this as a baseline.
- **User-based k-nearest neighbour**: for every user-item interaction, (u, i), we want to predict a rating for, the recommender systems finds the n users, most similar to u, and uses the weighted average of their ratings for i as prediction. The ratings are weighted by the similarity of the users.
- **Item-based k-nearest neighbour**: for every user-item interaction, (u, i), we want to predict a rating for, the recommender systems finds the n items, most similar to i, and uses the weighted average of their ratings for u as prediction. The ratings are weighted by the similarity of the items.

- **Singular Value Decomposition (SVD)**: this algorithm low-rank approximates the user-item ratings and uses the resulting latent approximation to estimate unknown ratings.
- **SVD++**: this technique improves upon SVD by adding user and item biases, taking into account that individual users and items can have structurally higher or lower ratings.
- **Co-clustering**: using k-means clustering, this algorithm assigns all users and items to clusters (containing either users or items) and co-clusters (containing both users and items) and uses the means of the clusters and co-clusters to calculate predictions.
- **SLIM**: this algorithm is a linear optimization model based on finding sparse neighbors for users and items.

To generate ratings predictions, we directly use the algorithm outputs. For finding rankings, we rank interactions according to the rating-based predictions of the algorithms in descending order. For implementing all algorithms but SLIM, we use the open-source Surprise framework [1]. We evaluate the SLIM algorithm using an open-source implementation [139].

5.2. EMISSION-METRICS

To evaluate the different algorithms in terms of emissions, a standardized method is required. Because of this, we created several metrics that can be used to benchmark the greenness of rating- and ranking-based systems.

5.2.1. RATING-BASED

Rating-based recommender systems predict the ratings users give to items. We argue that a purely green recommender system should predict ratings according to the following rules:

- For the greenest items (the items with the smallest amount of CO2), the highest ratings should be predicted.
- For the least green items (the items with the largest amount of CO2), the lowest ratings should be predicted.
- For the items with average greenness, an average rating should be predicted.

In order to evaluate the greenness of recommender systems, a metric is needed to score the rating outputs in a way that aligns with these requirements. This can be achieved by measuring the correlation between the predicted rating of an item an its corresponding greenness value: if there is a large positive correlation coefficient, the recommendation is regarded as 'desirable' (high rating/high greenness, low rating/low greenness).

[1] http://surpriselib.com/

Along this line of reasoning, if there is a negative correlation, the recommendation is regarded as 'undesirable'. Therefore, we propose the greenness-pearson correlation (GPC) as a measure for the greenness of rating-based algorithms. The metric is inspired by the Pearson correlation coefficient [140]. For a set of interactions, $\mathscr{I}$, resulting from a recommendation algorithm, the GPC is defined as:

$$\text{GPC} = \frac{\sum_{(u,i)\in\mathscr{I}} \left(\hat{r}_{u,i} - \bar{r}\right)\left(g_i - \bar{g}\right)}{\sqrt{\sum_{(u,i)\in\mathscr{I}} \left(\hat{r}_{u,i} - \bar{r}\right)^2}\sqrt{\sum_{i\in\mathscr{I}} \left(g_i - \bar{g}\right)^2}} \tag{5.1}$$

In which:

- $\mathscr{I}$ is the set of all interactions for which a rating has been predicted;
- i represents an item from interaction $\mathscr{I}$;
- u represents a user from interaction $\mathscr{I}$;
- $\hat{r}_{u,i}$ is the predicted rating of interaction $\mathscr{I}$ (i.e. the predicted rating for user u and item i);
- $\bar{r}$ is the average value of all predicted ratings for the interactions in $\mathscr{I}$;
- g_i is the greenness of item i. The greenness of an item can be calculated from its CO_2-eq. We present a technique for this in 5.3.1;
- $\bar{g}$ is the average greenness for all items that have interactions in $\mathscr{I}$

The numerator calculates and sums the correlations between the predictions and greenness values. This is done by comparing them to their corresponding mean values. Values close to the mean can be regarded as neutral, while values far from the mean can be regarded as either high or low. Values higher than the mean become positive, while lower values become negative. The sign of the term $\left(\hat{r}_{u,i} - \bar{r}\right)\left(g_i - \bar{g}\right)$ is therefore positive when there is a positive correlation (desirable) and negative when there is a negative correlation (undesirable). The magnitude of the correlation is determined by the relative differences from the mean. The denominator standardizes the metric, such that its range is $[-1, +1]$, where $+1$ represents a perfectly green algorithm and -1 a non-green algorithm. Note that this metric assumes that the predictions and the greenness' have the same scale. If this is not the case and the difference between the scales is large, the metric may be unsuitable for the overall greenness utility of a set of predictions, as either the rating or greenness values might cause each other to become insignificant.

The metric regards interactions with a low prediction and a low greenness as desirable as those with a high prediction and a high greenness. The same applies for negatively correlated values. We choose to do this, because we want recommender systems to not recommend non-green items as much as we want them to recommend green items. Therefore, this metric cannot directly be applied to scenarios where different behavior is required (e.g. when high predictions for non-green items are regarded worse than low predictions for green items).

5.2.2. RANKING-BASED

Ranking-based recommender systems predict the lists of items that may be most relevant to users. Therefore, we argue that the optimal green recommender system should rank items in a descending order of greenest; i.e., the greener items to be ranked first while the least green items last in the list. Based on this, we invent a metric, inspired by the discounted cumulative gain (DCG; Chap. 2). We call this new metric the greenness discounted cumulative gain (GDCG):

$$GDCG = \frac{1}{m}\sum_{u=1}^{m}\sum_{i\in I_u, v_i\leq L}\frac{gn_i}{\log_2(v_i+1)} \tag{5.2}$$

Where v_i is the rank of item i in set I_u. gn_i is calculated using the greenness of item i as follows:

$$gn_i = 2^{g_i} - 1 \tag{5.3}$$

Where g_i represents the greenness of item i. We chose to embed g_i as an exponent, because it allows us to place stronger emphasis on recommending green recipes higher in the list. This is desirable, as users tend to pay more attention to items they are recommend first [141]. Similarly to the NDCG, we normalize this metric to do meaningful comparisons. For this reason, we created the NGDCG (normalized greenness discounted cumulative gain):

$$NGDCG = \frac{GDCG}{IGDCG} \tag{5.4}$$

IGDCG represents the ideal normalized greenness discounted cumulative gain. It can be found by applying DCG on the list of recommendation, ordered by the ground-truth greenness of items. Naturally, rankings that are more ideal for a totally green recommender system, that is rankings that assign low ranks (high in the list) to green items, have *NGDCG* values close to 1. Non-ideal rankings have *NGDCG* values close to zero

We chose to base the GNDCG on the NDCG because it is an often-used method that has been proven to be effective.

5.3. EXPERIMENTAL SETUP

In this section, we describe the different steps of the conducted experiment: data processing, hyperparameter tuning and evaluation.

5.3.1. DATA PROCESSING

The used dataset is the one created in Chapter 4 with the following processing steps.

Scale transformations The metrics require ratings and greenness values. The greenness can be obtained from emissions by replacing all emission values with their inverse. The metrics also require the ratings and the greenness values in the same scale. Because the scale of the ratings is clearly defined and limited ([0,5]), the greenness values are transformed to a discrete scale in the range [0,5], similarly to the ratings. In order to achieve this, the greenness values are first transformed to a logarithmic scale. The distribution

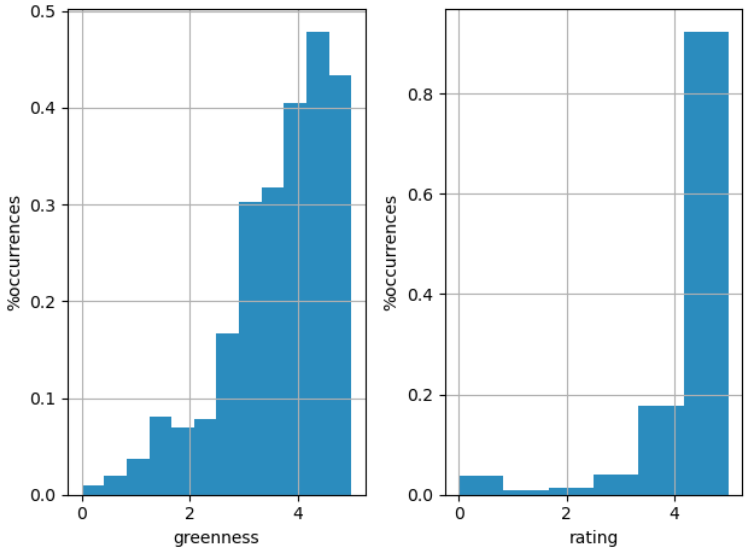


Figure 5.1: The distribution of the greenness values and rating values of the RecipeEmissions dataset.

of the greenness values is logarithmic, as we saw in Figure 4.2b. After this, the values are linearly transformed to be within the range of 0 to 5, similarly to the rating values. Formally, this procedure is:

$$g_i = \mathcal{Q}\left[\log\left(1+\frac{1}{c_i}\right)\right] \tag{5.5}$$

Where c_i is the CO_2-eq of item i and $\mathcal{Q}[\cdot]$ is the quantification operator to the range [0,5]. Because of the logarithmic distribution of the greenness values, the logarithmic scaling step is required. The logarithmic distribution causes that relatively more data points represent small greenness values, compared to large ones. Directly transforming the greenness values to fall between 0 and 5 (without the logarithmic step) would therefore remove important insights in the smaller greenness ranges, as these values would be "squashed" into a small interval. The distributions of the greenness and rating values of the RecipeEmissions dataset can be seen in Figure 5.1. For the greenness metrics to be as fair as possible, it is important that the two distributions are similar. The figure indicates that their distributions are logarithmically increasing, which shows that the distribution have similarities. The ratings, however, are more centered towards large values.

Splitting The data is split into a training set, a validation set and a test set, satisfying the following requirements:

- All sets should ideally be similar to the original dataset distribution in terms of greenness and ratings. Also the distributions describing the occurrences of specific numbers of ratings per item and ratings per user should be similar to the original set.

- We want to avoid bias towards strict cold-start users and items. Therefore, the test and validation sets should only contain users and items that are also in the training set.

The test set is created as follows to meet the second requirement:

- Initially, the training set is formed by the entire dataset of interactions between users and items. The test set is empty.
- While the test set consists of less than 40% of the interactions in the dataset, a random interaction is chosen from the subset of interactions. Only interactions for which both the user and the item is present at least two times in the dataset are considered. This is done to prevent cold-start users and items.

The validation set can be made by randomly splitting the test set into two equal subsets. Doing this will result in a training that contains 60% of the original data, and validation and test sets that each contain 20% of the original data. The size of 20% is chosen, as it is a well researched and often chosen size for validation and test sets. To increase the fairness of the experiment, we perform this procedure 5 times to obtain 5 different splits. The different splits will all be used to evaluate the algorithms.

The sets that result from following this procedure are compared to the entire original dataset in order to verify whether the similarity requirement holds. The distributions of a single training, validation and test set are shown in Figure 5.2. The greenness and rating distributions of the three sets are almost identical. They are also similar to the distribution of the original dataset. The similarities are important, because dissimilarities could introduce a bias in the results of this experiment. The other splits have almost identical distributions as shown in the figure.

5.3.2. HYPERPARAMETER TUNING

To do valid comparisons between the different algorithms, it is important that we tune the respective hyperparameters following a standardized approach. Therefore, we do a grid-search over a pre-specified search space and optimize with respect to minimizing the RMSE. The search spaces are chosen according to two basic heuristics. First of all, the parameters should be commonly used in literature. This gives justification to the choices made. However, it is infeasible to search all possible parameter combinations. Therefore, the number of parameters should also be chosen, such that it remains feasible to run the grid search. This is different for every algorithm, as it highly depends on the speed at which the algorithms can run.

In order to avoid this chapter to become unnecessarily lengthy and cluttered, the detailed description of the chosen hyperparameter search spaces and the best performing sets of parameters are described in Appendix B.

5.3.3. EVALUATION

Our evaluation is based on greenness and accuracy. A properly functioning green recommender system should be balanced between greenness and accuracy, since a recommender system that provides green but inaccurate recommendations has no practical

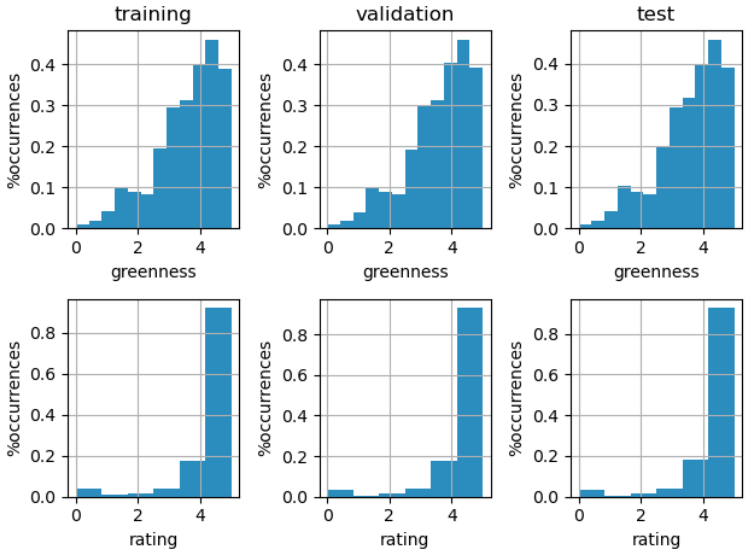


Figure 5.2: The greenness and rating distributions for the training, validation and test sets

use. We evaluate greenness using the GPC and GNDCG and accuracy using the root mean square error (RMSE) for the rating-based experiments and the NDCG for the ranking based experiments (see Chapter 2).

Figure 4.2b shows that the RecipeEmissions dataset has a bias towards large values, which influences the recommendation algorithms as well. Figure 5.2 further shows that similar biases are present in the training data, for both the ground truth ratings and the greenness values. This can be problematic, especially for the ranking-based evaluations, as they build rankings from predicted ratings. They order them and select the k highest ranking predictions for the recommendation list and k is generally between 5 and 50. At the same time, the test set contains almost 50000 interactions; thus when a bias causes many predictions to have the exact same high value, and k is small, it is highly likely that many, or even all, of the k recommendations have predictions of 5. This results in a biased NDCG that is higher than it actually is, ultimately, resulting into an unfair image of the GNDCG. This problem can never be completely mitigated as the effect of the bias in the dataset will remain. However, it is possible to reduce its effect. We do this by splitting the test set into subsets of 100 items each after the predictions have been generated. We rank the items in these subsets. For these rankings we calculate the accuracy and greenness scores. Lastly, we use the median of the scores of the subsets and use it as the final result.

Having sets of 100 interactions, instead of one set of almost 50000 greatly reduces the chance that (almost) all $top-k$ interactions will have 5 as predictions and greenness scores.

5.4. NUMERICAL RESULTS

This section presents the results of evaluating the different recommender systems.

5.4.1. RATING-BASED SOLUTIONS

Figure 5.3 shows the RMSE and GPC of the evaluated algorithms. Table 5.1 shows the corresponding numerical values including their standard deviation. In terms of RMSE, we observe that the SVD and SVD++ algorithms obtain the smallest error, followed by the SLIM algorithm. The item-based knn, user-based knn and co-clustering algorithms perform the worst, even worse than the global average baseline. This is likely caused by the high sparsity of the dataset. To illustrate this, the user-based knn algorithm directly uses a pre-specified number of similar neighbours to generate predictions for an interaction with a target item. Due to the sparsity, however, it can occur that not all neighbours of the user rated the target item. Then, the estimation is based on a small number of neighbours, or even a single neighbour. This reduces the accuracy of predictions, as the power of knn algorithms lies in their ability to abstract the combined data of many similar users. This also applies to the item-based knn algorithms. Specifically the user-based knn algorithm also suffers from the cold start problem. As the data is sparse, a large number of users have rated only a small number of items. Therefore, it is difficult to separate the neighbours that actually have similar interest from the neighbours who happen to have rated the same item. The co-clustering algorithm suffers from the same problems, as its clusters of similar users and items, and its clusters of user-item combinations likely contain users and items that lack shared interactions.

The matrix factorization algorithms SVD and SVD++ perform better, as they suffer less from the sparsity. Their factorization strategy allows them to generalize better over the entire interaction matrix, without being limited by individual neighbours. The same holds for the SLIM algorithm, which can be seen as a special case of matrix factorization. It should be noted that the global average baseline is relatively high. This is a result from the bias towards high ratings in the RecipeEmissions dataset. There is a large number of interactions with a rating of 5. Therefore, the global average (which is about 4.6 for all different training splits) is likely to be a good estimate of many ratings.

In terms of greenness, the algorithms are similar. All greenness values are close to 0, indicating that, in general, the algorithms are not green. There certainly are differences between the algorithms (i.e. ths SVD obtained a higher GPC than item-knn), but the differences are minimal, especially if we take the standard deviations into account. This is caused by the fact that the recommender systems are solely optimized with respect to accuracy. Therefore, they do not take greenness into account while generating recommendations. The GPC scores of the algorithms are therefore the result of accidental patterns in the data. Even though the differences are small, we can see that SVD is a dominating solution, because it has both the lowest RMSE and the highest GPC.

5.4.2. RANKING-BASED SOLUTIONS

Table 5.2 shows the NDCG values for the different algorithms and for different list lengths. We observe that the overall NDCG values are relatively large. As was explained before,

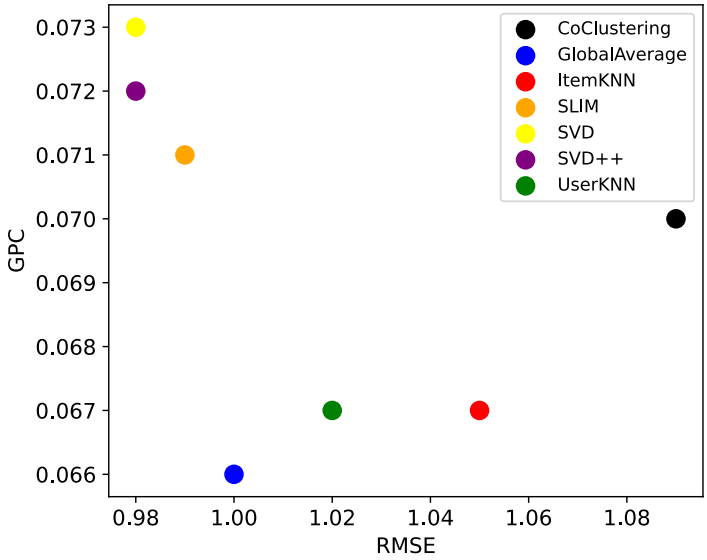


Figure 5.3: The RMSE scores of the different algorithms versus the GPC scores.

Table 5.1: The GPC and RMSE values of the benchmarked rating-based algorithms.

	Global Average	**item-knn (k=40)**	**user-knn (k=60)**	**SVD**	**SVD++**	**Co-Clustering**	**SLIM**
RMSE	1.00 (± 0.01)	1.05 (± 0.005)	1.02 (± 0.01)	0.98 (± 0.01)	0.98 (± 0.01)	1.09 (± 0.01)	0.99 (± 0.01)
GPC	0.066 (± 0.003)	0.067 (± 0.005)	0.067 (± 0.004)	0.073 (± 0.004)	0.072 (± 0.004)	0.070 (± 0.005)	0.071 (± 0.005)

this is partly caused by the bias towards ratings of 5. However, as this bias is part of the ground-truth data, it is similarly present in every recommender algorithm. Therefore, the results can validly be used to compare the different algorithms to each other.

By looking at the different list lengths, it can be seen that SVD performs the best for the small list length of 10 item, where it is closely followed by the SVD++ and SLIM algorithms. The worst performer is the GlobalAverage baseline, because it ranks interactions randomly as all interactions have the same prediction. For a list length of 20 and 50, SVD and SVD++ perform the best. Likely, the good performance of these algorithms can be ascribed to their ability to estimate ratings correctly (Sec. 5.4.1). This is directly correlated with their ability to rank, because of our ranking strategy. All methods perform better than the baseline. Interesting is the performance of the co-clustering algorithm. We saw that it suffered from a large RMSE as a rating-based solution, but, as a ranking-based solution, we observe that the method performs well. This indicates that the com-

bination on clusters and co-clusters is not very well able to estimate exact ratings, but it is properly capable of estimating the value of ratings compared to others.

Then, we can see an interesting pattern: when the length of the list grows, the NDCG scores never increase. Likely, this is again caused by the bias in the dataset. When the list length grows, the ratio of items with a prediction of 5 becomes smaller, causing the NDCG to become smaller automatically.

Table 5.2: NDCG values of the different ranking algorithms

	NDCG@10	NDCG@20	NDCG@50
GlobalAverage	0.87 (± 0.008)	0.86 (± 0.006)	0.86 (± 0.002)
ItemKNN	0.92 (± 0.006)	0.91 (± 0.0004)	0.91 (± 0.003)
UserKNN	0.89 (± 0.005)	0.89 (± 0.001)	0.87 (± 0.001)
SVD	0.97 (± 0.0009)	0.95 (± 0.003)	0.93 (± 0.003)
SVD++	0.96 (± 0.002)	0.95 (± 0.004)	0.93 (± 0.002)
CoClustering	0.95 (± 0.005)	0.94 (± 0.002)	0.92 (± 0.001)
SLIM	0.96 (± 0.003)	0.94 (± 0.004)	0.92 (± 0.003)

Table 5.3: GNDCG values of the different ranking algorithms

	GNDCG@10	GNDCG@20	GNDCG@50
GlobalAverage	0.46 (± 0.02)	0.51 (± 0.01)	0.62 (± 0.01)
ItemKNN	0.47 (± 0.007)	0.51 (± 0.001)	0.61 (± 0.001)
UserKNN	0.49 (± 0.006)	0.52 (± 0.005)	0.62 (± 0.001)
SVD	0.49 (± 0.004)	0.53 (± 0.002)	0.63 (± 0.003)
SVD++	0.49 (± 0.003)	0.52 (± 0.001)	0.62 (± 0.002)
CoClustering	0.48 (± 0.01)	0.53 (± 0.008)	0.62 (± 0.007)
SLIM	0.49 (± 0.005)	0.53 (± 0.004)	0.62 (± 0.003)

Table 5.3 shows the GNDCG values of the different algorithms. For a list length of 10, SVD, SVD++, CoClustering, SLIM and user-based knn are equally the most green. For a list length of 20, this place is for the SVD, CoClustering and SLIM algorithms. SVD scores the best for a list length of 50. However, the differences between all algorithm are again small, because, similar to the rating-based algorithms, the algorithms do not take greenness into account during ranking. Therefore, the small differences that we observe are unlikely to be the result of algorithmic specificity's.

Contrary to the NDCG, it can be seen that the GNDCG increases with the list length. This is a result the way the GNDCG works. As we explained in Section 5.2.2, the greenness is an exponent in the numerator. The result of this is that we prefer green-items to be high in the list of recommendation. The consequence of this is that very green items that have a low position in the ranking are still regarded as more valuable than non-green items with a slightly higher position. Taking into account that the algorithms do not use greenness for ranking, it becomes likely that if the size of the recommendation list grows, both the number of green and non-green items in the list grows as well. As

the lower positioned green items contribute more than the non-green higher-positioned items, the GDCG, and with that the NGDCG, grows.

5.5. DISCUSSION

We proposed a methodology to benchmark recommender systems, based on their accuracy and greenness. First, we developed metrics for measuring the greenness of both rating- and ranking-based recommender systems. After that, we designed an experimental setup to determine the accuracy and greenness of recommender systems. We then applied this setup to several existing recommender system algorithms. In this section, we first discuss the results that we obtained. Afterwards, we analyse the experimental setup in detail. Lastly, we conclude this chapter with some ideas for future research.

5.5.1. RESULTS

The experiment provided valuable insights with regard to the greenness of recommender systems. One of these insights is that most recommender system algorithms have very similar greenness values, but that they cannot be considered as green. This shows that the specific workings of the recommender systems have no significant effect on differences in greenness. The reason for this is that the recommender systems themselves do not take into account greenness when generating predictions. Instead, they are completely optimized for accuracy. Therefore, the obtained greenness scores are predominantly caused by the distribution of the data. Because of this, we cannot directly generalize the observations to other datasets. It is namely likely that the results are different if the data distributions change.

Even though the greenness of the algorithms does not strongly differ among them, we observed the presence of several dominating solutions. For the rating-based recommenders, SVD is dominating the other solutions, as it has the smallest errors and the largest greenness. The same holds for several ranking-based recommender systems, such as the SVD algorithm when $n = 10$. We attributed this performance to the ability of such algorithms to predict accurate ratings by abstracting over the entire dataset. The presence of these dominating solutions is important, as optimizing for two metrics can be a balancing act. When no dominating solutions are available, a trade-off has to be made with regard to which metric is deemed more important.

Also, the bias that is present in the data, both for high ratings and high greenness has several effects on the experiment. Most important, it causes the accuracy scores to be relatively high for the ranking-based approaches. This, however, does not affect the comparisons that have been made, as all algorithms are trained and evaluated on the same training and testing sets.

5.5.2. ANALYSIS OF EXPERIMENT

Metrics The experimental setup consists of many steps, all of which have advantages and disadvantages. An assumption that we made, while designing the GPC is that the greenness of predictions should be measured linearly. This is based on the belief that green items with high ratings are equally valuable as non-green items with low predictions and that non-green items with high ratings are as undesirable as green items with low ratings.

However, it might also be possible that a different method of scoring predictions could be more valuable to certain applications.

Contrary to this, we designed the GNDCG to logarithmically increase with the rank. The reasoning for this is that for many applications, it holds that users only take the first few recommendations of a ranking into account. Therefore, we believe that a green recommendation with a lower position in the list is logarithmically worse than a similarly green recommendation with a higher position in the list. Again, there also exist applications of recommender systems where this assumption does not hold. It might, for example, be possible that users of a car-buying platform look at many different options before they decide which car to purchase. In such scenarios, the logarithmic punishment could be replaced by a linear punishment.

Data Processing First, we transformed the greenness to a scale from 0 to 5. As stated in Chapter 4, to stimulate the utilization of green recommender systems in practice, it is important that data annotation has low costs. Precisely estimating CO_2 equivalent is difficult and requires significant resources. On the contrary, if greenness is indicated by a number between 0 and 5, it becomes simpler to annotate them. It is then namely only required that the greenness of items is correct, relative to other items. Greenness could then, for example, be estimated with a limited set of rules such as product material and travelled distance.

Naturally, there are some downsides to this step. By transforming values from a large scale to a far smaller scale, details in the data are lost. We partly mitigated the effects of this, by first transforming the greenness to a logarithmic scale. However, as was shown in Chapter 4, the greenness values are approximately logarithmically distributed, but not exactly. By treating the distribution to be exactly logarithmic in nature, we inevitably introduced noise in the data.

Hyperparameter tuning We tuned the hyperparameters of the recommender system algorithms via grid search. Grid search is a costly procedure, such that only a relatively small number of parameter values can be tested. It is therefore also likely that the found parameters are not the most optimal ones. However, as the decision to test these parameters was made based on existing literature, we expect these hyperparameters to be good.

5.5.3. FUTURE WORK

There are several things that would be interesting to research in the future. Related to the greenness metrics, we indicated that their required characteristics can differ per application. Therefore, it would be interesting to identify such requirements and develop metrics that adhere to them. The effect of the metrics on the numerical results could then be quantified, and benefits and drawbacks could be determined.

Moreover, it would be interesting to benchmark other algorithms with respect to greenness, especially to see if deep learning based solutions are more or less greener than non-deep learning ones. Also, a remaining question is how well the findings from this chapter hold for other datasets. Obtaining information about this could improve our understanding of greenness in recommender systems.

6

Nudging Towards Green Recommendations

This chapter contains the another core contribution of the thesis. It develops a technique for nudging users towards greener recommendations. In Section 6.1 we describe our technique and the experimental setup. Then, Section 6.2 provides the results of the experiment and Section 6.3 discusses the findings and future directions regarding nudging.

6.1. Experimental Setup

This section describes the setup of our experiment. We start by proposing a nudging strategy. After that, an overview of the settings of the experiment is provided briefly.

6.1.1. Nudging Strategy

The proposed nudging approach is a weighting method, similar to the techniques used in [71] and [142]. It is schematically shown in Figure 6.1. First, a recommender system algorithm generates rating predictions for a test-set without accounting for its greenness; i.e. similar to the analysis in Chapter 5. The predictions are then combined with the greenness of the corresponding items, resulting in a utility value $\mu_{u,i}$, which is calculated for every interaction individually as:

$$\mu_{u,i} = \alpha \hat{r}_{u,i} + (1 - \alpha) g_i \tag{6.1}$$

In this equation, $\mu_{u,i}$ is the utility of an interaction (u, i), $\hat{r}_{u,i}$ is the predicted rating for interaction (u, i) and g_i is the greenness of the item i. The parameter α can be set between 0 and 1 and determines the importance of both the prediction and greenness. An $\alpha \to 1$ causes the utility to be influenced more by the predictions and less by the greenness, while an $\alpha \to 0$ does the opposite. Naturally, when α is 1, the utility values are the same as the prediction values, and when α is 0, the same holds for the greenness values.

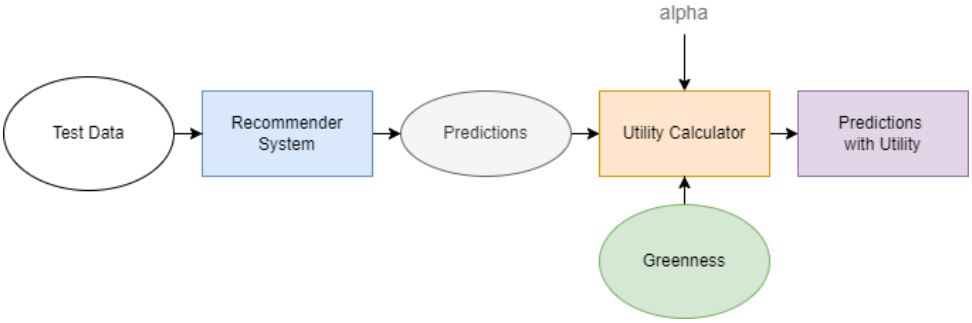


Figure 6.1: The proposed nudging approach. Predictions on the test data are made by a trained recommender system. The predictions and greenness are combined according to Equation **??** and a value for α, resulting in a utility value.

Because the strategy combines prediction and greenness into a combined utility, it can not be compared directly to other approaches using ratings. Therefore, we decide to evaluate this method only using rankings. The rankings can be found by ordering the results of this technique by their utility value in a descending way.

6.1.2. EXPERIMENT SETTINGS

We re-use the predictions generated from the benchmarking process in Chapter 5 (see Figure 6.1). We again create rankings by ordering the interactions by their rating predictions. This means that the techniques of data processing (5.3.1) and hyperparameter tuning (5.3.2) remain the same as in Chapter 5. Because of computational reasons, we decide to perform our experiment for lists of size 20. We measure performance using the *NDCG* and *GNDCG*. By doing this, we can directly compare this technique to the methods in Chapter 5. To obtain a complete overview of the influence of α on the predictions, we choose to experiment with values of alpha in the full range [0, 1], with a step size of 0.1 (i.e. [0, 0.1, 0.2 ... 0.8, 0.9, 1]).

6.2. EXPERIMENTAL RESULTS

The results the experiment are shown in Figure 6.2. The plot shows the NDCG and GNDCG for different values of α. There is a natural trade-off between the NDCG and GNDCG: if one increases, the other is likely to decrease. Figure 6.2 shows this trade-off by varying α. First, we observe that for most algorithms, the GNDCG grows logarithmically. For large values of α, the corresponding slope is steep. In other words, for large α values, the greenness of recommendations grows fast. At the same time, the NDCG decreases slowly, in an almost linear fashion. The slope of the decrease tends to become steeper when α becomes smaller. To illustrate this, when we compare the NDCG and GNDCG for the α values of 1 and 0.6, the GNDCG increases by 72.6%, while the NDCG decreases by only 2.7%. As a result, it becomes easier to determine the optimal trade-off for a green-recommender system in individual contexts. The behaviour of the trade-off is likely caused by the earlier observation that CO_2 equivalents of items have no major

impact on the ratings assigned to them (Fig. 4.5). It implies that greener alternatives for non-green items can be recommended to the users, without a significant change in the quality (rating) of a recommendation. For all algorithms, it can be seen that the GNDCG becomes 1 when α reaches 0. This is as expected, as the ranking of the interactions is completely determined by their greenness when α is 0.

Second, the GNDCG of the global average only grows between 1 and 0.9. There is a reason for this is that the predictions for the global average are constant across interactions, i.e. if the greenness is combined with the predictions to calculate the utility ($\alpha < 1$), the utility values are solely determined by the greenness. As a result, if the greenness is only taken slightly into account, it would determine the complete ranking. Even when the α change, the ranking does not change, because the different greenness values preserve the same relative order.

The SVD and SVD++ show a similar pattern, which indicates that the additional bias terms of SVD++ do not influence the trade-off. Both show a logarithmic increase for the GNDCG and a close-to-linear decrease for the NDCG. The same holds for the SLIM and co-clustering algorithms. The latter does show a slightly steeper increase in GNDCG between 1 and 0.9. This difference is unlikely to be the direct result of the way that the co-clustering algorithm generates predictions. It is more likely that is an incidental occurrence, caused by the underlying dataset.

The user-based knn and item-based knn algorithms show a different behaviour. The NDCG of the user-based knn algorithm decreases less for the same increase of GNDCG as the other algorithms. This is because the rankings of the algorithm are less accurate than those of the others. Therefore, when α is 0, the algorithm has already a significantly lower NDCG. A similar story applies to the item-based knn algorithm, but interestingly, the figure shows that for α values between 1 and 0.8, the NDCG of the algorithm even grows with the GNDCG. This is caused by the inability to generate highly accurate rankings, such that the green re-ranking that we perform does not only improve the greenness, but also the accuracy. It shows that integrating greenness does not necessarily have to lead to a decrease in accuracy.

Lastly, for the interval between α=1 and α=0.5, it holds that the slopes of the trade-offs are less steep than for the smaller values of α. This implies that the rankings change less, the more important greenness becomes in the re-ranking, showing that the rankings become dominated by the greenness, and that the predicted ratings slowly become less important.

6.3. DISCUSSION

In this chapter we introduced a method of improving the greenness of recommender systems. We introduced the the corresponding experimental setup, evaluated it and provided the results. In this section, we will discuss the results and their limitations further and we will provide several future research directions.

6.3.1. ANALYSIS, LIMITATIONS AND FUTURE WORK

Nudging people towards more sustainable decisions via recommender systems is a balancing act. It is possible to recommend only the greenest of items. However, such a

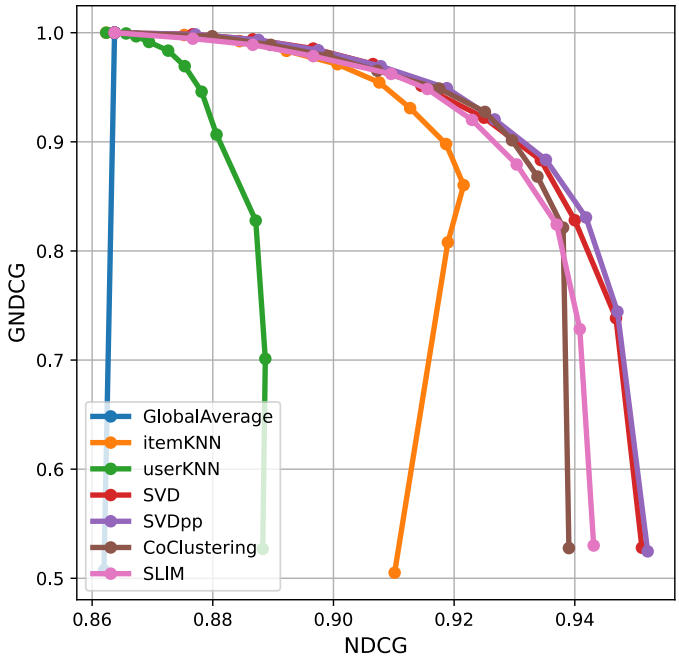


Figure 6.2: The results of nudging. The NDCG is plotted against the GNDCG. Every point represents a value of α in the range [0, 1], with a stepsize of 0.1

system is unlikely to be used in practice, as it would probably provide unsatisfying recommendations. Therefore, this would remove the incentive of users to use it, and eventually drive them away. Clearly, there is a trade-off between greenness and accuracy. The trade-off boils down to the question how much accuracy can be sacrificed for greenness.

In this regard, our results are very promising. We saw that the GNDCG grows logarithmically and that the NDCG decreases linearly when α is decreased. In practice, that means that we can sacrifice only little accuracy to get a substantial gain in greenness. This allows for nudging users to more sustainable decisions, without them having to receive less interesting recommendations. Potential future research should test the proposed method on other datasets to show whether the findings can be generalized across datasets and domains.

The downside of this method is that α needs to be chosen explicitly, which is related to the trade-off discussed earlier. Even though the steady increase in greenness for a minimal amount of accuracy loss likely eases this decision, there is no general method of choosing α. Its value depends on the application and the goal of the recommender system. There are several possible ways of setting this parameter. An expert with domain knowledge could, for example, determine the most suitable trade-off for a certain

context. However, the parameter can also directly be determined by the recommender system. As part of future research, one could use population-based multi-objective optimization for this, which can be applied to determine for which parameters the recommendations are Pareto optimal. Another solution is to let the parameter be decided directly by the user through an interface. In this regard, as part of future research, it would be interesting to investigate how users experience the selection of different α values in practice. Such research could be done via user studies and could verify the practical usefulness of green recommender systems.

7

DISCUSSION

In this chapter, we summarize our work and discuss the research question and subquestions. Moreover, we provide the limitations and the practical considerations. We conclude our research with possible future directions.

7.1. THESIS SUMMARY

In this thesis, we researched green recommender systems. In Chapter 1, we motivated our work and introduced the research questions. In Chapter 2 we, first, explained the background knowledge required for this thesis. We started with an explanation of recommender systems and several algorithms. Then, we presented different metrics for evaluating them. We covered both accuracy metrics and beyond accuracy metrics. In Chapter 3, we reviewed the existing literature that relates to our work. Concretely, we covered how machine learning is applied to mitigate climate change, how multi-objective recommender systems optimize recommendations for multiple goals simultaneously, and how recommender systems have been used to nudge users towards decisions.

In Chapter 4, we introduced the first dataset for green recommender systems. To achieve this, we annotated a dataset containing recipes, originally scraped from Food.com, with ingredient quantities and recipe emissions. We used both automated scraping as well as manual annotation techniques for this. We thoroughly discussed the resulting dataset and compared it to other often used recommender systems datasets. We showed that the properties of the datasets are similar, which supports the usability of the dataset.

In Chapter 5 we introduced a method for quantifying the greenness of recommendation algorithms. Next, we compared the performance of the global average, user-based k-nn, item-based k-nn, SVD, SVD++, co-clustering and SLIM algorithms. The results showed that there is no significant difference between the greenness of the different algorithms.

Finally, in Chapter 6 we proposed a re-ranking strategy for improving the greenness of ranked lists of recommendations. We applied the technique on the recommendations generated in Chapter 5 and found that significant improvements in greenness can be obtained without a significant decrease of accuracy.

7.2. ANSWERS TO THE POSED RESEARCH QUESTIONS

In this thesis, our main research question was: 'To what extent can greenness be integrated into recommender systems?'. In order to fully answer our question we posed the following subquestions;

> *(SQ1): Can we make a dataset to evaluate the greenness of recommender systems?*
>
> *(SQ2): Is there a difference in the greenness of current recommender system algorithms?*
>
> *(SQ3): How can we increase the greenness of recommendations?*

To answer the first subquestion, we developed the RecipeEmissions dataset in Chapter 4. The dataset is intended for researching green recommender systems and it is the first of its kind. We created the dataset by annotating a recipe dataset, originally scraped from Food.com, with ingredient quantities and CO_2 equivalents of recipes, whereby we automatically scraped as many CO_2-eqs as possible. From the values that could not be found automatically, we found as many CO_2-eqs as possible manually. The quantity values were also found by a combination of automated searches and manual annotation. For this, we were assisted by a team of annotators.

The resulting dataset contains 32.090 users, 5.595 recipes and 247.219 interactions. The interactions are expressed as numerical ratings between 0 and 5. All ingredients are annotated with an estimation of their CO_2-eq. We compared the dataset with several other often used recommender system datasets and the original Food.com dataset. This showed that the statistical properties of the RecipeEmissions dataset are similar to those of the other datasets. This indicates that the dataset can properly be used for research.

In practice, the dataset can be used to evaluate the greenness of recommender systems and to develop green recommender system algorithms. We intend the set to serve as a stepping stone that can enable researchers to advance the domain of green recommender systems.

We addressed the second subquestion in Chapter 5. We created two metrics to quantify the greenness of recommender system algorithms, one for rating-based algorithms, the GPC, and one for ranking-based ones, the GNDCG. Using the metrics, we benchmarked several existing recommendation algorithms: global average, user-based k-nn, item-based k-nn, SVD, SVD++, co-clustering and SLIM. The results showed that there were no significant differences between the greenness of the different algorithms. This is because the recommender systems ignore greenness when generating recommendations. Therefore, any differences in the results of the algorithms are the result of accidental patterns in the data. Moreover, by evaluating the accuracy of the algorithms, we found that the SVD, SVD++ and SLIM algorithms performed better than the clustering algorithms. Likely, this is a result of their ability to generalize better over the training data.

In practice, the proposed metrics can be used as a standardized method of evaluating the greenness of recommender systems. Additionally, the benchmarking results can act as a baseline to improve upon.

We have answered the third subquestion in Chapter 6. We proposed a method of improving greenness by re-ranking the results of the rating-based recommender algorithms according to a utility value. The utility value is a weighted sum between the predicted ratings and the greenness of items. The weighting embodies the trade-off there exists between the greenness of recommendations and their accuracy: a recommender system could be perfectly green, but when its recommendations are inaccurate, it is unlikely to be practical. We investigated this trade-off for different values of α. This showed that when we increase the weighted importance of the greenness (and consequently decrease that of the predictions), the greenness (GNDCG) of the recommendation rankings increases rapidly, while their accuracy (NDCG) decreases slowly. Hence, recommendations can be made significantly greener, without losing quality (accuracy). To illustrate this, for the SVD and $\alpha = 0.6$ we have that the GNDCG@20 is improved by 72.6% whereas the NCDG@20 is reduced by only 2.7% w.r.t. the benchmarking results.

The outcomes from our experiment are promising for practical applications. The trade-off between greenness and accuracy is arguably the most important component for the application of green recommender systems, as it determines whether users can effectively be nudged. Our results show that this trade-off would likely not be very difficult in practice (for the RecipeEmissions dataset), as large increases in greenness can be obtained with minimal losses in accuracy. They, therefore, indicate the ideal scenario, because users could be nudged towards decisions that allow them to greatly reduce their footprint, without noticing it. In other words, users would not have to handle the inconvenience of being provided worse recommendations, while at the same time their behaviour would be greener.

To answer our main research question, we find that it is to a great extent possible to integrate greenness into recommender systems, while retaining the usability of recommendations. However, we want to emphasize that this is the first work on green recommender systems. Therefore, there are some practical aspects that need to be considered.

7.3. LIMITATIONS, PRACTICAL CONSIDERATIONS AND FUTURE WORK

In this section we will discuss some of our practical considerations and limitations. As a concluding note we will, also, provide research directions for future work.

7.3.1. LIMITATIONS

Since, this is the first study done on green recommender systems, we were required to create our own dataset. We chose the domain of this to be food recipes. The E-commerce market, however, is larger than food and our results can, therefore, not directly be generalized to the entire domain.

Furthermore, we nudged recommendations to be greener by defining a utility that combined accuracy and greenness using a weighting parameter. Our method, however, is unable to find the best trade-off automatically. This is a limitation, since manually setting it is context dependent.

Lastly, we only considered the viability of green recommender systems recommendations on an algorithmic level. This means that we only quantitatively determined that

it is possible to make recommendations greener without losing their accuracy. However, this does not guarantee that this is also possible from a user-perspective. We can therefore not draw conclusions about the practical viability of green recommender systems.

7.3.2. PRACTICAL CONSIDERATIONS

The practical considerations of this research stem largely from the fact that this is the first work on integrating greenness into recommender systems. Consequently, we created every part related to greenness by ourselves: the RecipeEmissions dataset, the greenness metrics and the nudging strategy. Although we have taken a lot of aspects into account to make the research as thorough as possible, there are still several things that should be considered.

In terms of the dataset, we used annotators to find data. They used the internet to find information on ingredient quantities. It should be noted that using individuals to choose among information sources can be rather subjective in comparison to automated techniques. Therefore, this must be considered in the process of creating the RecipeEmissions dataset. Moreover, for finding the CO_2 equivalents, there is no extensive source available for all ingredients. We mitigated this by getting as much data as possible from a single source, but this was not possible for all ingredients. Therefore, as CO_2-eqs can differ per region, it is possible that the different sources resulted in relative inaccuracies.

In terms of the bias towards high rating and high greenness, explained in Chapter 4, it could have influenced the outcomes of the benchmarking and nudging experiments. For benchmarking, the bias likely increased the obtained accuracy of the algorithms, because the bias reduces the spread in the data. Having less spread, it becomes easier for algorithms to generate predictions that will be accurate for a large number of interactions. During nudging, we observed that the trade-off was very promising, as we obtained large increases in greenness, without major losses in accuracy. This has likely also been influenced by the combined biases of the accuracy and greenness. As the accuracy and greenness of many items is large, it namely becomes easier to find green replacements for non-green items that have a similar high rating.

7.3.3. FUTURE WORK

To advance research in the direction of green recommender systems, we argue that it is important to develop additional datasets. Using these, it will become possible to evaluate whether the results of our work are generalizable to other domains and whether new insights can be obtained. We think that the best way to achieve this, is by making the annotation process as simple as possible. In this thesis, we have chosen to estimate the CO_2-eq of items as precisely as possible. In practice, however, it should not matter what the exact CO_2-eq of a product is. Rather, it is important to understand how green items are compared to others. This tasks can be easier. Therefore, methods and frameworks should be researched that can be used to provide items with a score, relative to other items. Our approach for this was shown in Chapter 5. The metrics that we invented required the emissions of items to be on a scale of 0 to 5. This is an example of a technique that can make it less costly to create datasets: if the greenness of every item in a dataset should be rated with a number between 0 and 5, it is only required to understand how

green items are compared to others. For this, less domain knowledge is required and standardized frameworks can be created.

This does not mean, however, that datasets with precise CO_2-eq estimations are useless. On the contrary, they are important to understand and properly investigate the preference behavior of users. Therefore, there is a high need for creating such datasets too. To this end, we have created a crowdsourcing tool [1] that can be used to collect the user-preferences of cars in a publicly available dataset [2]. The dataset contains the CO_2 emission rates of the cars, but no user-ratings. With the platform, we intend to produce another dataset to build upon in future research.

Moreover, for nudging we determined the required accuracy-greenness trade-off manually by setting a parameter. While this could be a good solution for domains where the trade-off is clear, it might be a less desirable solution for others. There is much potential in methods that automatically determine the trade-off and these should therefore be researched. A possible way of achieving this could be by using a population-based multi-objective recommender systems to determine Pareto-optimal recommendations.

Lastly, we think that the practical usability of green recommender systems should be researched. After all, the algorithmic metrics can show improvements, but if the usability is low, there is not much practical use. Therefore, as part of future research, we argue that user-studies have to be performed that can show how users experience green recommender systems. This could not only provide insights into the value of green recommender systems in general, but it could also show how users prefer the accuracy-greenness trade-off. This knowledge could then again be used to improve the algorithmic aspects of green recommenders.

Clearly, green recommender systems have much potential. Therefore, we are convinced that, with the help of our work and future works, this concept could be the basis for a new research paradigm. And, that it can evidently contribute to a future with no rising temperatures, flooded cities and destroyed ecosystems.

[1] http://ratingcroudsourcing.ewi.tudelft.nl/
[2] https://www.kaggle.com/datasets/debajyotipodder/co2-emission-by-vehicles

Appendices

A

APPENDIX

This appendix provides details on the estimation of the weight of the non-standard quantity measures. The tables contain categories, representations and the (total average) amount of grams

Table A.1: Categories, their representations and their corresponding weights for 'pint' metric. The total value is the average of the total weight in grams.

Category	**Represented by**	**Weight (g)**
liquids	water	473.18
sugars and sweeteners	raw sugar	414.02
flours	all-purpose flower	250.31
oils and fats	olive oil	434.38
spices	cinnamon	264.98
nuts	almonds	217.66
fruits and vegetables	onion	104.10
Total		308.38

Table A.2: Categories, their representations and their corresponding weights for 'pinch' metric. The total value is the average of the total weight in grams.

Category	**Represented by**	**Weight (g)**
spices	salt	0.36
Total		0.36

Table A.3: Categories, their representations and their corresponding weights for 'cup' metric. The total value is the average of the total weight in grams.

Category	**Represented by**	**Weight (g)**
liquids	water	236.59
sugars and sweeteners	raw sugar	227.36
flours	all-purpose flower	125.16
oils and fats	olive oil	217.66
spices	cinnamon	132.49
nuts	almonds	108.83
fruits and vegetables	onion	150
Total		171.16

Table A.4: Categories, their representations and their corresponding weights for 'teaspoon' metric. The total value is the average of the total weight in grams.

Category	**Represented by**	**Weight (g)**
liquids	water	4.93
sugars and sweeteners	raw sugar	4.74
flours	all-purpose flower	2.61
oils and fats	olive oil	4.53
spices	cinnamon	2.76
Total		3.91

Table A.5: Categories, their representations and their corresponding weights for 'tablespoon' metric. The total value is the average of the total weight in grams.

Category	**Represented by**	**Weight (g)**
liquids	water	14.79
sugars and sweeteners	raw sugar	14.21
flours	all-purpose flower	7.82
oils and fats	olive oil	13.6
spices	cinnamon	8.28
Total		11.74

B

APPENDIX

This appendix provides the hyperparameter selection on which a grid-search is performed to optimize the algorithms' performance. Bold values are the selected, most performing, parameters. All different data splits required the same parameters.

Table B.1: Parameters included in the grid search for optimizing item-based knn. The values in bold form the chosen selections.

k	20	**40**	60	80

Table B.2: Parameters included in the grid search for optimizing user-based knn. The values in bold form the chosen selections.

k	20	40	**60**	80

Table B.3: Parameters included in the grid search for optimizing SVD. The values in bold form the chosen selections.

Number of factors	**Learning Rate**	**Regularization**	**Epochs**
20	0.1	**0.1**	20
50	0.01	0.01	**50**
100	**0.001**	0.001	80
150	0.0001	0.0001	

Table B.4: Parameters included in the grid search for optimizing SVD++. The values in bold form the chosen selections.

Number of factors	Learning Rate	Regularization	Epochs
20	0.1	**0.1**	20
50	0.01	0.01	**50**
100	**0.001**	0.001	80
150	0.0001	0.0001	

Table B.5: Parameters included in the grid search for optimizing co-clustering. The values in bold form the chosen selections.

Clusters	Epochs
3	20
6	**50**
12	80
24	

Table B.6: Parameters included in the grid search for optimizing SLIM. The values in bold form the chosen selections.

L1 regularization	L2 regularization
0.005	0.005
0.05	**0.05**
0.5	0.5

BIBLIOGRAPHY

[1] M. Salam, T. Noguchi, *et al.*, "Impact of human activities on carbon dioxide (co2) emissions: A statistical analysis," *Environmentalist*, vol. 25, no. 1, pp. 19–30, 2005.

[2] *The surprising case for stronger e-commerce growth.* [Online]. Available: https://www.morganstanley.com/ideas/global-ecommerce-growth-forecast-2022.

[3] *Our carbon footprint.* [Online]. Available: https://sustainability.aboutamazon.com/environment/carbon-footprint.

[4] *Global gridded model of carbon footprints.* [Online]. Available: https://www.citycarbonfootprints.info/.

[5] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, "Recommender systems," *Physics reports*, vol. 519, no. 1, pp. 1–49, 2012.

[6] Y. Jiang, J. Shang, and Y. Liu, "Maximizing customer satisfaction through an online recommendation system: A novel associative classification model," *Decision Support Systems*, vol. 48, no. 3, pp. 470–479, 2010.

[7] P.-Y. Chen, S.-y. Wu, and J. Yoon, "The impact of online recommendations and consumer feedback on sales," 2004.

[8] I. MacKenzie, C. Meyer, and S. Noble, *How retailers can keep up with consumers*, Feb. 2018. [Online]. Available: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers.

[9] B. P. Majumder, S. Li, J. Ni, and J. McAuley, "Generating personalized recipes from historical user preferences," *arXiv preprint arXiv:1909.00105*, 2019.

[10] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[11] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[12] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[13] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender systems handbook*, Springer, 2011, pp. 217–253.

[14] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[15] R. H. Singh, S. Maurya, T. Tripathi, T. Narula, and G. Srivastav, "Movie recommendation system using cosine similarity and knn," *International Journal of Engineering and Advanced Technology*, vol. 9, no. 5, pp. 556–559, 2020.

[16] B. Li and L. Han, "Distance weighted cosine similarity measure for text classification," in *International conference on intelligent data engineering and automated learning*, Springer, 2013, pp. 611–618.

[17] R. Mehta and K. Rana, "A review on matrix factorization techniques in recommender systems," in *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, IEEE, 2017, pp. 269–274.

[18] X. Zhou, J. He, G. Huang, and Y. Zhang, "Svd-based incremental approaches for recommender systems," *Journal of Computer and System Sciences*, vol. 81, no. 4, pp. 717–733, 2015.

[19] S. Funk, *Netflix update: Try this at home*, Dec. 2006. [Online]. Available: https://sifter.org/simon/journal/20061211.html.

[20] C. Teflioudi, F. Makari, and R. Gemulla, "Distributed matrix completion," in *2012 ieee 12th international conference on data mining*, IEEE, 2012, pp. 655–664.

[21] S. Gower, "Netflix prize and svd," *University of Puget Sound*, 2014.

[22] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, IEEE, 2005, 4–pp.

[23] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *2011 IEEE 11th international conference on data mining*, IEEE, 2011, pp. 497–506.

[24] C. C. Aggarwal *et al.*, *Recommender systems*. Springer, 2016, vol. 1.

[25] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks.," *Journal of Machine Learning Research*, vol. 10, no. 12, 2009.

[26] B. Smyth and P. McClave, "Similarity vs. diversity," in *International conference on case-based reasoning*, Springer, 2001, pp. 347–361.

[27] C. Yu, L. V. Lakshmanan, and S. Amer-Yahia, "Recommendation diversification using explanations," in *2009 IEEE 25th International Conference on Data Engineering*, IEEE, 2009, pp. 1299–1302.

[28] M. Kaminskas and D. Bridge, "Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 7, no. 1, pp. 1–42, 2016.

[29] S. J. Kamble and M. R. Kounte, "Machine learning approach on traffic congestion monitoring system in internet of vehicles," *Procedia Computer Science*, vol. 171, pp. 2235–2241, 2020.

[30] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang, "Solving the apparent diversity-accuracy dilemma of recommender systems," *Proceedings of the National Academy of Sciences*, vol. 107, no. 10, pp. 4511–4515, 2010.

[31] D. Rolnick, P. L. Donti, L. H. Kaack, *et al.*, "Tackling climate change with machine learning," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–96, 2022.

[32] I. C. Change *et al.*, "Mitigation of climate change," *Contribution of working group III to the fifth assessment report of the intergovernmental panel on climate change*, vol. 1454, p. 147, 2014.

[33] R. Ahmed, V. Sreeram, Y. Mishra, and M. Arif, "A review and evaluation of the state-of-the-art in pv solar power forecasting: Techniques and optimization," *Renewable and Sustainable Energy Reviews*, vol. 124, p. 109 792, 2020.

[34] J. Li, J. K. Ward, J. Tong, L. Collins, and G. Platt, "Machine learning for solar irradiance forecasting of photovoltaic system," *Renewable energy*, vol. 90, pp. 542–553, 2016.

[35] J. Mathe, N. Miolane, N. Sebastien, and J. Lequeux, "Pvnet: A lrcn architecture for spatio-temporal photovoltaic powerforecasting from numerical weather prediction," *arXiv preprint arXiv:1902.01453*, 2019.

[36] J. Maldonado-Correa, J. Solano, and M. Rojas-Moncayo, "Wind power forecasting: A systematic literature review," *Wind Engineering*, vol. 45, no. 2, pp. 413–426, 2021.

[37] G. Giebel and G. Kariniotakis, "Wind power forecasting—a review of the state of the art," *Renewable energy forecasting*, pp. 59–109, 2017.

[38] L. Xiao, J. Wang, Y. Dong, and J. Wu, "Combined forecasting models for wind energy forecasting: A case study in china," *Renewable and Sustainable Energy Reviews*, vol. 44, pp. 271–288, 2015.

[39] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *International Journal of Forecasting*, vol. 32, no. 3, pp. 914–938, 2016.

[40] B. Yildiz, J. I. Bilbao, and A. B. Sproul, "A review and analysis of regression and machine learning models on commercial building electricity load forecasting," *Renewable and Sustainable Energy Reviews*, vol. 73, pp. 1104–1122, 2017.

[41] H. K. Alfares and M. Nazeeruddin, "Electric load forecasting: Literature survey and classification of methods," *International journal of systems science*, vol. 33, no. 1, pp. 23–34, 2002.

[42] N. Moehle, E. Busseti, S. Boyd, and M. Wytock, "Dynamic energy management," in *Large Scale Optimization in Supply Chains and Smart Manufacturing*, Springer, 2019, pp. 69–126.

[43] Y. Liu, Q. Zhou, and G. Cui, "Machine learning boosting the development of advanced lithium batteries," *Small Methods*, vol. 5, no. 8, p. 2 100 442, 2021.

[44] C. P. Gomes, J. Bai, Y. Xue, *et al.*, "Crystal: A multi-agent ai system for automated mapping of materials' crystal structures," *MRS Communications*, vol. 9, no. 2, pp. 600–608, 2019.

[45] D. D. A. Ellman, "The reference electrification model: A computer model for planning rural electricity access," Ph.D. dissertation, Massachusetts Institute of Technology, 2015.

[46] H. Ren, R. Stewart, J. Song, V. Kuleshov, and S. Ermon, "Learning with weak supervision from physics and data-driven constraints," *AI Magazine*, vol. 39, no. 1, pp. 27–38, 2018.

[47] *Global warming of 1.5 ºc - ipcc*, 2018. [Online]. Available: https://www.ipcc.ch/sr15/.

[48] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2018.

[49] Y. Li and C. Shahabi, "A brief overview of machine learning methods for short-term traffic forecasting and future directions," *Sigspatial Special*, vol. 10, no. 1, pp. 3–9, 2018.

[50] C. Sun, N. Azari, and C. Turakhia, "Gallery: A machine learning model management system at uber.," in *EDBT*, 2020, pp. 474–485.

[51] T. Anagnostopoulos, "A predictive vehicle ride sharing recommendation system for smart cities commuting," *Smart Cities*, vol. 4, no. 1, pp. 177–191, 2021.

[52] A. Y. Suatmadi, F. Creutzig, and I. M. Otto, "On-demand motorcycle taxis improve mobility, not sustainability," *Case Studies on Transport Policy*, vol. 7, no. 2, pp. 218–229, 2019.

[53] J. A. Badra, F. Khaled, M. Tang, *et al.*, "Engine combustion system optimization using computational fluid dynamics and machine learning: A methodological approach," *Journal of Energy Resources Technology*, vol. 143, no. 2, 2021.

[54] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[55] V. Marano, S. Onori, Y. Guezennec, G. Rizzoni, and N. Madella, "Lithium-ion batteries life estimation for plug-in hybrid electric vehicles," in *2009 IEEE vehicle power and propulsion conference*, IEEE, 2009, pp. 536–543.

[56] M. Berecibar, *Machine-learning techniques used to accurately predict battery life*, 2019.

[57] M. Aykol, C. B. Gopal, A. Anapolsky, *et al.*, "Perspective—combining physics and machine learning to predict battery lifetime," *Journal of The Electrochemical Society*, vol. 168, no. 3, p. 030 525, 2021.

[58] J. Hagenauer and M. Helbich, "A comparative study of machine learning classifiers for modeling travel mode choice," *Expert Systems with Applications*, vol. 78, pp. 273–282, 2017.

[59] D. Nam, H. Kim, J. Cho, and R. Jayakrishnan, "A model based on deep learning for predicting travel mode choice," in *Proceedings of the transportation research board 96th annual meeting transportation research board, Washington, DC, USA*, 2017, pp. 8–12.

[60] A. Yazdizadeh, Z. Patterson, and B. Farooq, "Semi-supervised gans to infer travel modes in gps trajectories," *Journal of Big Data Analytics in Transportation*, vol. 3, no. 3, pp. 201–211, 2021.

[61] T. Seo, T. Kusakabe, H. Gotoh, and Y. Asakura, "Interactive online machine learning approach for activity-travel survey," *Transportation Research Part B: Methodological*, vol. 123, pp. 362–373, 2019.

[62] L. Wu, B. Yang, and P. Jing, "Travel mode detection based on gps raw data collected by smartphones: A systematic review of the existing methodologies," *Information*, vol. 7, no. 4, p. 67, 2016.

[63] C. Rudloff and M. Ray, "Detecting travel modes and profiling commuter habits solely based on gps data," Tech. Rep., 2010.

[64] M. Maghrebi, A. Abbasi, and S. T. Waller, "Transportation application of social media: Travel mode extraction," in *2016 IEEE 19th International Conference on intelligent transportation systems (ITSC)*, IEEE, 2016, pp. 1648–1653.

[65] H. Li, D. Parikh, Q. He, *et al.*, "Improving rail network velocity: A machine learning approach to predictive maintenance," *Transportation Research Part C: Emerging Technologies*, vol. 45, pp. 17–26, 2014.

[66] A. Singla, M. Santoni, G. Bartók, P. Mukerji, M. Meenen, and A. Krause, "Incentivizing users for balancing bike sharing systems," in *Twenty-Ninth AAAI conference on artificial intelligence*, 2015.

[67] R. Regue and W. Recker, "Proactive vehicle routing with inferred demand to solve the bikesharing rebalancing problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 72, pp. 192–209, 2014.

[68] W.-H. Lin and J. Zeng, "Experimental study of real-time bus arrival time prediction with gps data," *Transportation Research Record*, vol. 1666, no. 1, pp. 101–109, 1999.

[69] M. Altinkaya and M. Zontul, "Urban bus arrival time prediction: A review of computational models," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 2, no. 4, pp. 164–169, 2013.

[70] D. Panovski and T. Zaharia, "Long and short-term bus arrival time prediction with traffic density matrix," *IEEE Access*, vol. 8, pp. 226 267–226 284, 2020.

[71] E. Bothos, D. Apostolou, and G. Mentzas, "Recommender systems for nudging commuters towards eco-friendly decisions," *Intelligent Decision Technologies*, vol. 9, no. 3, pp. 295–306, 2015.

[72] S. Seyedzadeh, F. P. Rahimian, I. Glesk, and M. Roper, "Machine learning for estimation of building energy consumption and performance: A review," *Visualization in Engineering*, vol. 6, no. 1, pp. 1–20, 2018.

[73] J. Kreider, D. Claridge, P. Curtiss, R. Dodier, J. Haberl, and M. Krarti, "Building energy use prediction and system identification using recurrent neural networks," 1995.

[74] J. Kolter, S. Batra, and A. Ng, "Energy disaggregation via discriminative sparse coding," *Advances in neural information processing systems*, vol. 23, 2010.

[75] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," in *Artificial intelligence and statistics*, PMLR, 2012, pp. 1472–1482.

[76] P. Rashidi and D. J. Cook, "Keeping the resident in the loop: Adapting the smart home to the user," *IEEE Transactions on systems, man, and cybernetics-part A: systems and humans*, vol. 39, no. 5, pp. 949–959, 2009.

[77] T. Vafeiadis, S. Zikos, G. Stavropoulos, *et al.*, "Machine learning based occupancy detection via the use of smart meters," in *2017 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, IEEE, 2017, pp. 6–12.

[78] J. Kolter and J. Ferreira, "A large-scale study on predicting and contextualizing building energy usage," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, 2011, pp. 1349–1356.

[79] C. E. Kontokosta and C. Tull, "A data-driven predictive model of city-scale energy use in buildings," *Applied energy*, vol. 197, pp. 303–317, 2017.

[80] M. Aftab, C. Chen, C.-K. Chau, and T. Rahwan, "Automatic hvac control with real-time occupancy recognition and simulation-guided model predictive control in low-cost embedded system," *Energy and Buildings*, vol. 154, pp. 141–156, 2017.

[81] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate model predictive building control via machine learning," *Applied Energy*, vol. 218, pp. 199–216, 2018.

[82] X. Zhang, G. Hug, J. Z. Kolter, and I. Harjunkoski, "Model predictive control of industrial loads and energy storage for demand response," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, IEEE, 2016, pp. 1–5.

[83] D. A. Narciso and F. Martins, "Application of machine learning tools for energy efficiency in industry: A review," *Energy Reports*, vol. 6, pp. 1181–1199, 2020.

[84] G. Tsoumakas, "A survey of machine learning techniques for food sales prediction," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 441–447, 2019.

[85] A. O. Akyuz, M. Uysal, B. A. Bulbul, and M. O. Uysal, "Ensemble approach for time series analysis in demand forecasting: Ensemble learning," in *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, IEEE, 2017, pp. 7–12.

[86] Y. Zheng and D. X. Wang, "A survey of recommender systems with multi-objective optimization," *Neurocomputing*, vol. 474, pp. 141–153, 2022.

[87] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[88] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: New insights," *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[89] I. Y. Kim and O. L. De Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," *Structural and multidisciplinary optimization*, vol. 29, no. 2, pp. 149–158, 2005.

[90] M. Javadi, M. Lotfi, G. J. Osório, *et al.*, "A multi-objective model for home energy management system self-scheduling using the epsilon-constraint method," in *2020 IEEE 14th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG)*, IEEE, vol. 1, 2020, pp. 175–180.

[91] Y. Du, L. Xie, J. Liu, Y. Wang, Y. Xu, and S. Wang, "Multi-objective optimization of reverse osmosis networks by lexicographic optimization and augmented epsilon constraint method," *Desalination*, vol. 333, no. 1, pp. 66–81, 2014.

[92] A. Charnes, W. W. Cooper, and R. O. Ferguson, "Optimal estimation of executive compensation by linear programming," *Management science*, vol. 1, no. 2, pp. 138–151, 1955.

[93] R. E. Steuer and E.-U. Choo, "An interactive weighted tchebycheff procedure for multiple objective programming," *Mathematical programming*, vol. 26, no. 3, pp. 326–344, 1983.

[94] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 1–23, 1993.

[95] J. Kennedy, "Swarm intelligence," in *Handbook of nature-inspired and innovative computing*, Springer, 2006, pp. 187–219.

[96] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[97] C. M. Fonseca and P. J. Fleming, "Multiobjective genetic algorithms," in *IEE colloquium on genetic algorithms for control systems engineering*, Iet, 1993, pp. 6–1.

[98] L. Xiao, Z. Min, Z. Yongfeng, G. Zhaoquan, L. Yiqun, and M. Shaoping, "Fairness-aware group recommendation with pareto-efficiency," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017, pp. 107–115.

[99] Y. Wu, N. Yang, and H. Luo, "Unified group recommendation towards multiple criteria," in *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, Springer, 2019, pp. 137–151.

[100] Y. Zheng and A. Pu, "Utility-based multi-stakeholder recommendations by multi-objective optimization," in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, IEEE, 2018, pp. 128–135.

[101] N. R. Kermany, W. Zhao, J. Yang, J. Wu, and L. Pizzato, "An ethical multi-stakeholder recommender system based on evolutionary multi-objective optimization," in *2020 IEEE International Conference on Services Computing (SCC)*, IEEE, 2020, pp. 478–480.

[102] M. Unger, P. Li, M. C. Cohen, B. Brost, and A. Tuzhilin, "Deep multi-objective multi-stakeholder music recommendation," *NYU Stern School of Business Forthcoming*, 2021.

[103] T. Di Noia, J. Rosati, P. Tomeo, and E. Di Sciascio, "Adaptive multi-attribute diversity for recommender systems," *Information Sciences*, vol. 382, pp. 234–253, 2017.

[104] A. Paul, Z. Wu, K. Liu, and S. Gong, "Robust multi-objective visual bayesian personalized ranking for multimedia recommendation," *Applied Intelligence*, vol. 52, no. 4, pp. 3499–3510, 2022.

[105] S. Patil, D. Banerjee, and S. Sural, "A graph theoretic approach for multi-objective budget constrained capsule wardrobe recommendation," *ACM Transactions on Information Systems (TOIS)*, vol. 40, no. 1, pp. 1–33, 2021.

[106] R. Xie, Y. Liu, S. Zhang, R. Wang, F. Xia, and L. Lin, "Personalized approximate pareto-efficient recommendation," in *Proceedings of the Web Conference 2021*, 2021, pp. 3839–3849.

[107] A. Jain, P. K. Singh, and J. Dhar, "Multi-objective item evaluation for diverse as well as novel item recommendations," *Expert Systems with Applications*, vol. 139, p. 112 857, 2020.

[108] L. Cui, P. Ou, X. Fu, Z. Wen, and N. Lu, "A novel multi-objective evolutionary algorithm for recommendation systems," *Journal of Parallel and Distributed Computing*, vol. 103, pp. 53–63, 2017.

[109] Y. Zuo, M. Gong, J. Zeng, L. Ma, and L. Jiao, "Personalized recommendation based on evolutionary multi-objective optimization [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 10, no. 1, pp. 52–62, 2015.

[110] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani, "Pareto-efficient hybridization for multi-objective recommender systems," in *Proceedings of the sixth ACM conference on Recommender systems*, 2012, pp. 19–26.

[111] L. Zhang, X. Zhang, F. Cheng, X. Sun, and H. Zhao, "Personalized recommendation for crowdfunding platform: A multi-objective approach," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2019, pp. 3316–3324.

[112] Y. Cao, B. J. Smucker, and T. J. Robinson, "On using the hypervolume indicator to compare pareto fronts: Applications to multi-criteria optimal experimental design," *Journal of Statistical Planning and Inference*, vol. 160, pp. 60–74, 2015.

[113] M. Ge, F. Ricci, and D. Massimo, "Health-aware food recommender system," in *Proceedings of the 9th ACM Conference on Recommender Systems*, 2015, pp. 333–334.

[114] R. H. Thaler, C. R. Sunstein, and J. P. Balz, *Choice architecture*. Princeton University Press Princeton, NJ, 2013, vol. 2013.

[115] T. M. Marteau, D. Ogilvie, M. Roland, M. Suhrcke, and M. P. Kelly, "Judging nudging: Can nudging improve population health?" *Bmj*, vol. 342, 2011.

[116] M. Quigley, "Nudging for health: On public policy and designing choice architecture," *Medical law review*, vol. 21, no. 4, pp. 588–621, 2013.

[117] M. Weinmann, C. Schneider, and J. v. Brocke, "Digital nudging," *Business & Information Systems Engineering*, vol. 58, no. 6, pp. 433–436, 2016.

[118] M. Jesse and D. Jannach, "Digital nudging with recommender systems: Survey and future directions," *Computers in Human Behavior Reports*, vol. 3, p. 100 052, 2021.

[119] J. Turland, L. Coventry, D. Jeske, P. Briggs, and A. van Moorsel, "Nudging towards security: Developing an application for wireless network selection for android phones," in *Proceedings of the 2015 British HCI conference*, 2015, pp. 193–201.

[120] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[121] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th international conference on World Wide Web*, 2005, pp. 22–32.

[122] E. Bontempi, "A new approach for evaluating the sustainability of raw materials substitution based on embodied energy and the co2 footprint," *Journal of Cleaner Production*, vol. 162, pp. 162–169, 2017.

[123] V. Šerkinić, M. Majić Renjo, and V. Ucović, "Co2 footprint for distribution oil immersed transformers according to iso 14067: 2018," *Journal of Energy: Energija*, vol. 69, no. 3, pp. 0–0, 2020.

[124] W. Min, S. Jiang, and R. Jain, "Food recommendation: Framework, existing solutions, and challenges," *IEEE Transactions on Multimedia*, vol. 22, no. 10, pp. 2659–2671, 2019.

[125] *Glossary:carbon dioxide equivalent*. [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary3ACarbon_dioxide_equivalent.

[126] S. Netherlands, *Co2 equivalents*, May 2020. [Online]. Available: https://www.cbs.nl/en-gb/news/2020/19/uitstoot-broeikasgassen-3-procent-lager-in-2019/co2-equivalents.

[127] *Inch calculator*, Feb. 2020. [Online]. Available: https://www.inchcalculator.com/.

[128] A. Marie, *Thyme benefits + side effects: Your ethical guide*, Jul. 2022. [Online]. Available: https://www.healabel.com/thyme/.

[129] A. Marie, *Allspice benefits + side effects: Your ethical guide*, Jul. 2022. [Online]. Available: https://www.healabel.com/Allspice/.

[130] M. Yuttitham, S. H. Gheewala, and A. Chidthaisong, "Carbon footprint of sugar produced from sugarcane in eastern thailand," *Journal of Cleaner Production*, vol. 19, no. 17-18, pp. 2119–2127, 2011.

[131] M. Sevenster and F. de Jong, "A sustainable dairy sector," *Global, regional and life-cycle facts and figures on greenhouse-gas emissions. Delft, the Netherlands: CE Delft*, 2008.

[132] A. Dalla Riva, J. Burek, D. Kim, G. Thoma, M. Cassandro, and M. De Marchi, "Environmental life cycle assessment of italian mozzarella cheese: Hotspots and improvement opportunities," *Journal of dairy science*, vol. 100, no. 10, pp. 7933–7952, 2017.

[133] F. Fantozzi, P. Bartocci, and P. Fantozzi, "Life cycle assessment in the vinegar sector," in *Advances in Vinegar Production*, CRC Press, 2019, pp. 469–490.

[134] R. Parajuli, M. D. Matlock, and G. Thoma, "Cradle to grave environmental impact evaluation of the consumption of potato and tomato products," *Science of The Total Environment*, vol. 758, p. 143 662, 2021.

[135] R. Taylor, H. Omed, and G. Edwards-Jones, "The greenhouse emissions footprint of free-range eggs," *Poultry science*, vol. 93, no. 1, pp. 231–237, 2014.

[136] V. Vasilaki, E. Katsou, S. Ponsá, and J. Colón, "Water and carbon footprint of selected dairy products: A case study in catalonia," *Journal of cleaner production*, vol. 139, pp. 504–516, 2016.

[137] N. Pelletier, M. Ibarburu, and H. Xin, "A carbon footprint analysis of egg production and processing supply chains in the midwestern united states," *Journal of Cleaner Production*, vol. 54, pp. 108–114, 2013.

[138] N. Espinoza-Orias, H. Stichnothe, and A. Azapagic, "The carbon footprint of bread," *The International Journal of Life Cycle Assessment*, vol. 16, no. 4, pp. 351–365, 2011.

[139] X. Ning, A. N. Nikolakopoulos, Z. Shui, M. Sharma, and G. Karypis. "SLIM Library for Recommender Systems." (2019), [Online]. Available: https://github.com/KarypisLab/SLIM.

[140] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*, Springer, 2009, pp. 1–4.

[141] C. Burges, T. Shaked, E. Renshaw, *et al.*, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 89–96.

[142] K. Roher and D. Richardson, "A proposed recommender system for eliciting software sustainability requirements," in *2013 2nd international workshop on user evaluations for software engineering researchers (USER)*, IEEE, 2013, pp. 16–19.