A Recommendation System for Participatory Budgeting

Gil Leibiker Nimrod Talmon
Ben-Gurion University, Israel

ABSTRACT

In participatory budgeting, voters specify their preferences over a set of projects of different costs and the goal is to select a subset of these projects that satisfy some total cost upper bound while taking into account the preferences of the voters. To lower the cognitive burden on voters, and to increase voter participation in PB processes, we propose an approach based on a machine learning techniques as recommendation system and binary classification that queries voters for partial ballots and estimates their completion. We develop several concrete algorithms and evaluate them – based on real-world instances – wrt. their ability to correctly approximate voter ballots as well as the overall outcome of the process.

KEYWORDS

Computational Social Choice, Attention-Aware Social Choice, Participatory Budgeting, Machine Learning, Recommendation System

ACM Reference Format:
Gil Leibiker Nimrod Talmon, Ben-Gurion University, Israel. 2023. A Recommendation System for Participatory Budgeting. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 11 pages.

1 INTRODUCTION

Social Choice Theory is a branch of economics that studies collective decision-making processes and the preferences of individual agents. It aims to determine how a majority can make decisions based on the preferences of a group, while still respecting each individual involved. The theory explores various voting systems and ways to structure voting procedures in order to more accurately represent the will of the majority. Social Choice Theory is an important research area as it seeks to understand collective behavior, and can be applied to numerous fields such as political decision-making, economic welfare, and game theory. In essence, Social Choice Theory provides a framework to better understanding how our decisions impact others, and how we can make decisions that are both equitable and efficient [5, 13, 17]

Computational social choice is a field of research that combines computer science, game theory and social choice theory to develop algorithms for collective decision making. It examines how groups of people can use computers to make decisions that are fair, efficient and based on inputs from all involved parties. Computational social choice can be used in a variety of real-world scenarios such as voting systems, resource allocation, market design and even communications networks [5].

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

A sub-field of Computational social choice is Participatory Budgeting (PB). PB is a democratic process in which community members decide how to spend a portion of a public or private budget. PB allows members of a community to make their voices heard regarding funding decisions affecting a range of fields such as education, health and environment. It gives them the power to compose project proposals and to express their preferences over the budget allocation [1–3, 7]. In our study we concentrate on the most popular variant of PB, namely approval-based combinatorial PB. That is, given a set of projects – each with its cost – each voter expresses its preferences over the projects by selecting a subset of the projects that she "approves of", and the result of the process of aggregating such voter preferences is a subset of the projects whose total cost does not go over some given budget upper bound [2, 3].

In this research we consider processes of PB and concentrate on the information overload problem, i.e., the cognitive burden of voters participating in such processes. Essentially, the information overload problem appears when too much information interferes with decision making [11, 20]. To mitigate the information overload problem we propose several solutions based on techniques from machine learning and recommendation systems.

Recommendation systems(RS) are a powerful tool in machine learning, utilizing algorithms to process large datasets and make personalized predictions for users [14, 20]. By learning user preferences and usage patterns, RS can provide users with tailored and relevant recommendations for products, services, or content [11, 20]. For example, a movie RS could suggest films based on past user ratings, genre preferences, and even social connections. Similarly, a restaurant RS could suggest eateries based on location, cuisine type, and price range. Moreover, RS are being applied in many other areas such as music, books, and shopping, making them an important part of machine learning. The information overload problem is deeply related to the research on RS [4, 20, 24]. It may effect the decision making of participants in such systems in various aspects, in particular as users may find it challenging to choose the most appropriate choices out of the huge amount and variety of content that the internet offers. The goal of a RS is to support users in various domains of decision making processes, such as what items to purchase, which movies to watch, or which books to read [14, 20, 22]. In particular, the research on RS, being well-established by now, offers valuable tools for online users to cope with information overload and help them in making better choices.

In this paper we discuss the problem of information overload in PB where voters have to consider a large number of projects, leading to a time-consuming and attention-demanding task [8, 27]. We propose a solution that involves an application that uses machine learning and RS to estimate a voter's preferences for the remaining projects based on their opinions about a few projects. This approach could save voters time and allow them to express better preferences for the proposals they do consider. The paper highlights the benefits

of this system, which could lead to a more efficient and effective PB process.

1.1 Paper Structure

After providing an introduction, in Section 1, we go on to discuss some preliminaries (in Section 2) regarding participatory budgeting processes, machine learning methods and our algorithmic tasks. Then, we continue – in Section 3.1 – by describing the different problem variants we consider and its corresponding algorithms. In sections 4, 5 and 6 we describe the data sets that we examined our experiments on, discuss about the evaluation methods and describe the various experiments we conducted. In Section 7 we present and discuss our experimental results. We conclude in Section 8, in which we discuss possible avenues for future research.

2 PRELIMINARIES

We discuss on the binary classification technique and recommendation systems (Section 2.1), the standard model of combinatorial participatory budgeting (PB; Section 2.2), the model of PB with partial ballots that we use here (Section 2.3) and the algorithmic task we dealt with (Section 2.4).

2.1 Binary Classification and Recommendation Systems

We divide this section to two parts: Binary Classification and Recommendation systems.

2.1.1 Binary classification. Binary classification is a supervised machine learning technique in which a model is trained to predict one of two possible outcomes, often represented as "0" or "1" [18]. This type of classification is commonly used in a wide range of applications. The primary goal of binary classification is to identify which of the two classes an input sample belongs to, based on a set of features or attributes [18]. The process of training a binary classifier involves providing the model with labeled examples of each class, and then adjusting the model's parameters to minimize the classification error on the training set. Once the model is trained, it can then be used to make predictions on new, unseen samples. One popular algorithm for binary classification is XGBoost (eXtreme Gradient Boosting). This algorithm is an ensemble learning technique that has been widely used in binary classification tasks [9, 26]. It is an implementation of gradient boosting, a powerful ensemble method that combines the predictions of multiple weak learners to produce a more accurate prediction [9, 26]. XGBoost is specifically designed for decision tree-based models, and it has been observed to outperform other popular machine learning algorithms [9]. Overall, XGBoost is a powerful and widely-used tool for binary classification tasks, and it is well-suited for datasets with complex, non-linear relationships [9, 26]

2.1.2 Recommendation Systems. RS are tools that are based on machine learning techniques. Their purpose is to gather and analyze information regarding the preferences of users wrt. a set of items in order to deliver for each user a set of *recommended items* that the user hopefully will find interesting to interact with [14, 20, 24]. RS are information filtering systems that process user's and item's information to provide personalized prediction on the interaction between an user and an item, meaning the prediction of the rating of the item that may be given by the user (such a rating can be represented in various ranges [14, 20, 24]). RS techniques are extensively used in the e-commerce industry [4, 20], where there is a business aspect for how good the recommendations are.

	Item 1	Item 2	Item 3	Item 4
User 1	1	2	5	?
User 2	3	?	5	?
User 3	?	4	2	5
User 4	4	2	?	1

Figure 1: Rating matrix example. Each cell represents the rating that a user gives to a particular item.

The information based on which such RS operate can be acquired explicitly by collecting users' ratings. Since the rising of the popularity of social networks and other platforms, there is a huge corpus of available data that RS tools can use for their analysis and operation [14, 20, 22]. Another way for gathering user information is by a more implicit way, i.e., by analyzing user behavior on aspects such as web searches and other forms of usage history [4, 14]. In our research we focused on Collaborative filtering (CF) and Hybrid RS.

Collaborative Filtering. The collaborative filtering (CF) approach plays an important role in RS tools. It operates by analyzing relationships between users and items in order to identify new user-item interactions [15, 20, 21]. CF relies on an old, automatic act of people sharing opinion between them in order to make better decisions. For instance, a person may decide to go to a certain restaurant after hearing good things about it from several friends. Corresponding, hearing very bad reviews regarding a new movie from colleagues may result in deciding to watch another movie instead. Essentially, the CF technique operates in a similar fashion, however it allows to consider such opinions on a much larger scale [20, 21].

In a CF scenario, a user-item matrix is constructed, where each cell in the matrix represents the rating that a user gives to a particular item. Formally, a rating consists of the association of two things – user and item, one way to visualize ratings is as a matrix where each row represents a user, each column represents an item, and the number at the intersection of a row and a column represents the user's rating value as shown in Figure 1. The absence of a rating score at this intersection indicates that user has not yet rated the item [20, 21].

One primary field of CF is latent factor models [15, 20, 21]. Latent factor models are a popular technique in RS for capturing the underlying preferences and attributes of users and items [15, 21]. The idea behind latent factor models is that the observed ratings are generated by a low-dimensional latent space, where each user and item is represented by a set of latent factors [15, 21]. The learned latent factors can then be used to make personalized recommendations to users. Latent factor models can handle the sparsity problem that arises in large user-item matrices and can make recommendations even for users with limited ratings. One latent factor model is the Matrix Factorization (MF), MF can then be used to factorize this large user-item matrix into two smaller matrices: a user matrix and an item matrix [15]. The user matrix captures the latent preferences

of the users, while the item matrix captures the latent attributes of the items [15, 16]. These factorized matrices can then be used to make personalized recommendations to users by finding the items that are most similar to the items they have rated highly in the past. By using MF, CF can handle the sparsity problem that arises in large user-item matrices, where many cells are empty, and make recommendations even for users with limited ratings [15, 16, 20].

Hybrid approach - Factorization Machines. Hybrid RS combine two or more recommendation techniques to that allow a RS to make accurate and reliable recommendations with fewer of the drawbacks of any individual one [6]. The hybrid approach combines the best of both methods to provide a more holistic approach that can better identify user preference and personalize the recommendations to each individual user [6, 20]. It helps to overcome the limitations of each method such as cold start, scalability and sparsity by combining the strengths of both and producing better recommendations than any individual method could provide [16, 20].

Factorization Machines (FM) is a hybrid approach that increasingly gaining traction due to their ability to generate high quality and personalized recommendations [19]. FM are learning algorithms which combine the strengths of linear models and MF methods to learn from sparse datasets [19]. This enables them to capture intricate user-item interactions and the associated latent factors, resulting in improved accuracy and personalization for recommending products, movies, and other items to individual users.

FM models possess several advantages over existing recommendation algorithms, mainly due to their unique structure [19]. Unlike other models, which rely on fixed feature representations that are often tailored to one type of dataset, FM models are capable of automatically learning complex feature representations derived from interactions between users and items [19, 20]. This essentially allows them to capture the underlying features that influence users' preferences in an effective manner [19]. In addition, FM models are highly efficient in terms of computational cost, as they require minimal pre-processing and require only a small dataset to produce accurate results [19, 20].

2.2 Participatory Budgeting

In our research we focus on *combinatorial PB*, as it is the most studied and applied model of PB [2]. Below we describe the ingredients of an instance of combinatorial PB. Note that combinatorial PB is a formal generalization of multiwinner elections [1].

2.2.1 Projects. The set of projects is denoted by $P = \{p_1, \ldots, p_m\}$. Each project $p \in P$ has its cost C_p (usually, the resource taking into account for cost is money [2]). Note that model is indivisible, and, in particular, each project can be either fully implemented (at its given cost) or not implemented at all. In combinatorial PB projects are either fully implemented or not implemented at all.

2.2.2 Voters. The set of voters is denoted by $V = \{v_1, \ldots, v_n\}$, where each voter submits her preferences on the potential projects. We consider approval-based processes, in which the ballot of a voter corresponds to a **subset** of the set of projects (i.e., voter v_i submits an approval ballot, also denoted by v_i, such that $v_i \subseteq P$).

2.2.3 Popularity and Consensus. Given a set V of voters with their approval ballots over a set P of projects, it is useful to consider the *approval score* of each project as well as the *consensus degree* of each project.

Definition 2.1 (Approval scores). $score(p) = |\{v_i \in V : p \in v_i\}|$.

The *approval score* of a project p, denoted by $score(p)$, is the number of voters that approve it. We furthermore define σ to be the set P of projects, sorted in decreasing order of their scores; in particular, σ_1 is the project that has the most votes while σ_m is the least popular project.

While σ orders the projects according to their popularity, γ, defined next, orders the projects according to how much the voters are in consensus regarding them.

The *consensus level* of a project p, denoted by $consensus(p)$, is the absolute difference between the number of voters approving p and the number of voters disapproving p;

Definition 2.2 (Consensus levels). $consensus(p) = abs(|\{v_i \in V : p \in v_i|, |\{v_i \in V : p \notin v_i\}|)$.

Note that, in particular, both a project that is approved by all voters as well as a project that is disapproved by all voters have consensus level n. Correspondingly, we define γ to be the set P of projects, sorted in decreasing order of their consensus levels; in particular, γ_m is the "most controversial" project.

2.2.4 Budget upper limit and outcome. The *budget limit* is denoted by B. The *outcome* (also referred to as the *winning bundle*) of a PB process is a set of projects $p^\star \subseteq P$ where $\sum_{p \in p^\star} C_p \leq B$ (i.e., that respects the budget limit).

2.2.5 Voting rules and greedy approval. A *voting rule* is a function taking an instance of PB and returning a winning bundle. The most popular voting rule for PB can be described as a *greedy approval* [25]; intuitively, it considers the projects according to their approval scores, decreasingly, and keeps funding projects until the budget is exhausted.

Formally, greedy approval proceeds in iterations and maintains a *temporary budget* B' and a *temporary winning bundle* S'. Initially, $B' := B$ and $S' = \emptyset$. In the ith iteration, $i \in [n]$, the project σ_i is considered. If $C_{\sigma_i} \leq B'$, then the amount C_{σ_i} is decreased from B' and σ_i is added to S'. After the nth iteration, S' contains the winning bundle.

Example 2.3. Consider a toy example with $P = \{p_1, p_2, p_3\}$, where $C_{p_1} = C_{p_2} = 1$, $C_{p_3} = 2$, and $B = 3$; and with $V = \{v_1, v_2, v_3\}$ such that $v_1 = \{p_1, p_2\}$, $v_2 = \{p_1, p_3\}$, and $v_3 = \{p_2\}$.

Note that $score(p_1) = score(p_2) = 2$ and $score(p_3) = 1$. Thus, in the first and second iterations of greedy approval, p_1 and p_2 will be already selected to be included in the winning bundle; and, so, in the third iteration of greedy approval, in which p_3 is considered, there is not enough budget left to fund it, so it will be skipped. The winning bundle is thus $\{p_1, p_2\}$.

2.3 Participatory Budgeting with Partial Ballots

Recall that we are interested in situations in which **not** all voters provide complete ballots. In a ballot with m candidates a partial vote is case where voter gives preferences only over j candidates,

where $1 \leq j < m$ [12]. Our goal is to create and study algorithms that, when given a partial ballot as an input, will produce output as all voters having completely filled out their ballots. Specifically, we assume that each voter is associated with a set of approved candidates, a set of disapproved candidates, and a set of candidates where the voter's stand is unknown. Formally, voter $v \in \{v_1, \ldots, v_n\}$ has an approval set $A_v \subset P$, a disapproval set $D_v \subset P$ and a hidden set denoted by $H_v \subset P$. Note that $A_v + D_v + H_v = P$. We denote that the merge of approved and disapproved candidates sets is an exposed set $A_v \cup D_v = E_v, E_v \in P$.

Example 2.4. Consider an instance of PB with partial ballots with 3 projects: p_1, p_2, and p_3; and with 2 voters: v_1 and v_2. Let it be that v_1 provides a full ballot in which she approves all projects, while v_2 provides a partial ballot in which she approves p_1, disapproved p_2 and does not provide her preference regarding p_3. We denote such a setting as follows: $P = \{p_1, p_2, p_3\}$, $V = \{v_1, v_2\}$, $E_1 = \{p_1, p_2, p_3\}$, $A_1 = \{p_1, p_2, p_3\}$, $D_1 = \emptyset$, $H_1 = \emptyset$, $E_2 = \{p_1, p_2\}$, $A_2 = \{p_1\}$, $D_2 = \{p_2\}$, $H_2 = \{p_3\}$.

2.4 Algorithmic Tasks

We consider three slightly different algorithmic tasks, corresponding to different ways in which voters may interact with our envisioned application, or to different assumptions regarding the partial ballots. In all three variants, we assume some *ideal instance* of PB containing n voters, all of which provide full ballots; denote this instance by *ideal*. Intuitively, in the algorithmic tasks we consider, our goal is to estimate *ideal*, however the algorithms we design do not have access to *ideal* but only to a PB instance with partial ballots that *agrees* with *ideal* (a PB instance with partial ballots is said to *agree* with some PB instance with full ballots if all the provided preferences agree; intuitively, if it is possible to complete the partial instance to be equal to the ideal instance).

Example 2.5. Consider an ideal instance I with projects $P = \{p_1, p_2, p_3, p_4\}$ and one voter v_1 where her exposed set $E_1 = \{p_2, p_4\}$. Consider a PB instance with partial ballots, denoted by $I1$, with one voter v_1' that provides a partial ballot, and let $A_1 = \{p_2\}$ and $D_1 = p_1$. Then, $I1$ agrees with I.

To describe our algorithmic tasks, consider some ideal instance I with a set V of n voters. Then, consider an instance of PB with partial ballots $I1$ that is constructed by first partitioning two groups of voters *Learning Voters - LV* and *Target Voters - TV*. Intuitively, LV are voters that already provided their full ballots and TV are voters that provides partial ballots. Consider the voters in TV and set them initially so that $E = \emptyset$; i.e., such that they do not provide any preference. The overarching goal of our algorithms is to estimate I from $I1$. Each of the three algorithmic tasks we consider differentiate by the allowed operations that the algorithm can perform on $I1$ to reach this goal, as described next:

Random setup - An algorithm for the *random setup* operates as follows: first, for each voter $v \in TV$, the algorithm chooses k projects uniformly at random and moves them into E, in a way that $I1$ **agrees** with I (possibly different E for each voter $v \in TV$). Then, the algorithm has to **predict** the rest the ballot - H for each voter $v \in TV$.

Offline setup - An algorithm for the *offline setup* operates as follows: first, for each voter $v \in TV$, the algorithm chooses k projects of its choice (possibly by taking into consideration the ballots of LV, intuitively assuming some similarity between voters in LV and TV in I) and moves them into E, in a way that $I1$ **agrees** with I. Then, the algorithm has to **predict** the rest of the ballots - H for each voter $v \in TV$.

Online setup - An algorithm for the *online setup* operates as follows: first, for each voter $v \in TV$, the algorithm proceeds in k iterations, where in each iteration the algorithm can choose 1 project of its choice (possibly by taking into consideration the ballots of LV as well as the results of the previous iterations, intuitively assuming some similarity between voters in TV and LV in I and using the revealed preferences of the previous iterations) and moves it into E, in a way that $I1$ **agrees** with I. Then, the algorithm has to **predict** the rest of the ballots - H for each voter $v \in TV$. A more intuitive exposition of these algorithms tasks could be the following: imagine n voters providing their approval ballots where, pictorially, each voter has m cards, one card for each project, on which the voter writes their approval/disapproval preference. Then, some voters (those in LV) show all their cards; while others (those in TV) initially do not show their cards at all. In the *random setting*, each such voter shows k cards at random to the algorithm; in the *offline setting*, the algorithm can choose, for each such voter, a set of k cards that the algorithm wants to see; and in the *online setting*, the algorithm can **iteratively** asks for a first card, a second card, until a kth card, to see from each such voter. Finally, the algorithm shall predict **all** of the hidden cards. Illustrations of the different settings can be seen in Figure 2.

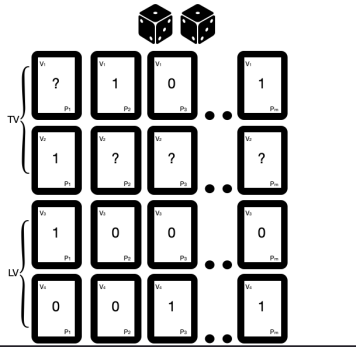
3 SOLUTION ARCHITECTURE

In this section we discuss the architecture of our solution, which consists of three modules: sampling module, prediction module and voting rule implementation. A graphical representation of our solution architecture is given in Figure 3.

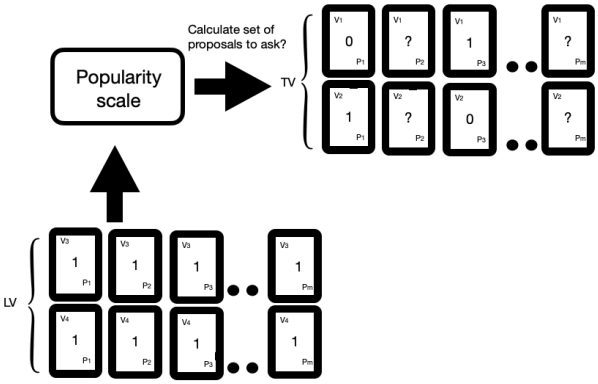
As mentioned, we propose a computational solution for PB with partial ballots, where some voters do not complete ballots. As we want to control the amount of information to gather from voters, we define a level of partiality as how sparse is our PB and how we divide the level of partiality to LV and TV.

3.0.1 Partiality level, Sampling Degree and LV Degree. The *Partiality level* of a PB denoted by *sample degree* as the percentage of data collecting from the voters participate in PB. We denote LV Degree as the level of collected data that associated with LV. As the level of sampling Degree increases, voters are requested to provide more data and as the level of LV Degree increases the $|LV|$ is increases.

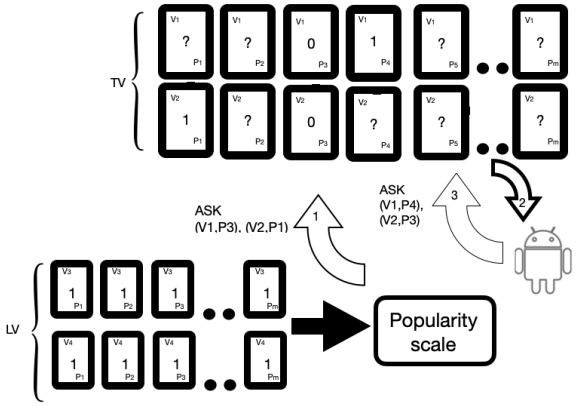
Example 3.1. Where Sampling Degree is 1 it means that the PB is complete, all voters filled their ballots. When Sampling Degree is 0.5 and LV Degree is 1 it means that 50% of votes are known and all of them are part of full individual ballots. When Sampling Degree is 0.5 and LV Degree is 0.1 it means that 50% of votes are known and 10% of them are part of full individual ballots and 90% are equally divided between TV.



(a) The random setting - the algorithm chooses randomly k projects from each voter to exposed.



(b) Offline Setting - the algorithm chooses a set of k projects that it asks for each voter to expose.



(c) Online Setting - the algorithm chooses iteratively k projects that it asks for each voter to expose.

Figure 2: Algorithmic Tasks.

3.1 Algorithmic Solutions

We describe several algorithmic solutions for each of the setups we consider, namely for the random setup, the offline setup, and the online setup.

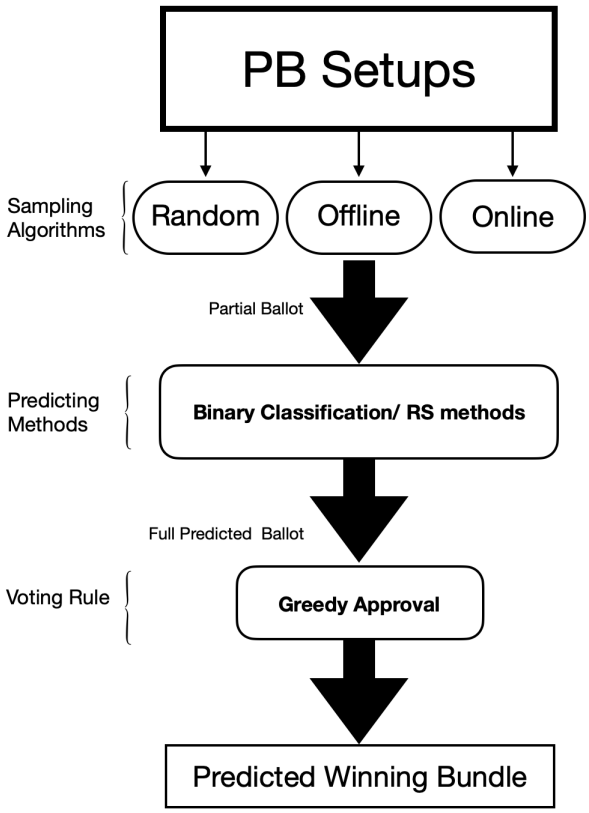


Figure 3: Architecture: starting with sampling algorithm - as the way we collect preferences from voters- gain partial ballot; then filling the missing votes by applying prediction module; implying voting rule to output the predicted winning bundle.

3.1.1 Algorithms for the Random Setup. Recall that an algorithm for this setup cannot choose the k projects for which the voters in TV provide preferences for; thus, the only operation that such an algorithm does is to predict the remaining preferences of the voters in TV, based on the preferences of voters in LV and the preferences in $E_{TV} = (E_1, \ldots, E_n)$ by prediction methods discussed in Section 2.1.

3.1.2 Algorithms for the Offline Setup. Recall that an algorithm for this setup first has to choose, for each $v \in TV$, a set k of projects to "reveal" (to set as E); and, only then, to predict the remaining preferences of the voters in TV, based on the preferences of voters in LV and the preferences in $E_{TV} = (E_1, \ldots, E_n)$ by prediction methods discussed in Section 2.1.

- Revealing by popularity
  Here the algorithm chooses the top k popular projects of the voters in LV and set the E of each voter $v \in TV$ to include exactly those. (I.e., it sets $E = \{\sigma_1, \ldots, \sigma_k\}$ for each $v \in TV$.)
- Revealing by consensus

Here the algorithm chooses the k projects most in consensus among the voters in LV and set the E of each voter $v \in TV$ to include exactly those. (I.e., it sets $E = \{\gamma_1, \ldots, \gamma_k\}$ for each $v \in TV$.)

- Revealing by controversiality
  Here the algorithm chooses the k projects **least** in consensus among the voters in LV and set the E of each voter $v \in TV$ to include exactly those. (I.e., it sets $E = \{\gamma_{m-k}, \ldots, \gamma_m\}$ for each $v \in TV$.)
  The intuition for this procedure is that the projects least in consensus correspond to those projects that are the hardest to predict.

3.1.3 Algorithms for the Online Setup. Recall that an algorithm for this setup first has to choose, for each $v \in TV$, a set k of projects to "reveal" (to set as E); and, only then, to predict the remaining preferences of the voters in TV, based on the preferences of voters in LV and the preferences in E. But, in contrast to the offline setting, the algorithm can choose these k projects **adaptively**. We consider one way of choosing the k projects to reveal, adaptively, namely: adaptive controversial. The algorithm proceeds in k iterations, where in each iteration it computes the most controversial project. (Note that, indeed, this procedure is similar to the *revealing by controversiality* procedure of the offline setting, however here it is done iteratively and adaptively.) After using the adaptive controversial procedure, we predict the remaining hidden preferences, using prediction methods discussed in Section 2.1.

3.1.4 Voting Rule. After we perform the predicting module we hold a full ballot, we now chose greedy approval as a voting rule. As described in Section 2.2.5 for this voting system, each voter is able to vote for as many candidates as they choose, but cannot vote for the same candidate twice. The algorithm then begins with the candidate who has the most number of votes and selects them. Each subsequent candidate is chosen until the total number of votes reaches the desired threshold. The aim of this system is to maximize the amount of "approval" given to the chosen candidates by the voters [25]. Hence, we output the predicted winning bundle with respect to a given budget.

4 DATA SETS

We have used a real-world data sets coming for a public repository, namely Pabulib [10], a public repository for data regarding real-world instances of participatory budgeting, mainly from several European cities. In particular, we have used the data concerning the PB process in different districts from Warsaw, Poland. Out of this open library we collected ten PB that took place from 2020 to 2023. Table 1 details the amount of voters, amount of projects and budget for each PB. Table 2 details the attributes of PB's voters and projects. Each PB instance consist voters and projects (the projects correspond to different proposals to improve the development of Warsaw in several domains such as, e.g., education, environment protection, public transportation, etc.). The proposals are also aimed for different segments of the city population, e.g. children, adults, seniors, and people with disabilities. Note that a certain proposal can consist multiple topics and population segments. In all PBs the percentage of voter's approved project proposals is naturally small, around 10% of project proposals were approved in every PB, the exact percentage is detailed in Table 1.

Table 1: Description of used real-world datasets.

PB	Voters	Projects	Budget	Approved proposals[%]
Wola 2022	9256	94	524020	9.5
Ursynow 2022	6672	107	5614506	10.7
Wola 2021	8647	107	465432	9.8
Praga-Polnoc 2022	2614	90	2432952	12.5
Praga-Poludnie 2022	10424	96	6643832	10.6
Wawer 2021	4662	100	2493341	9.1
Bielany 2020	8003	108	4321791	7.7
Wawer 2022	5045	111	2807253	7.8
Wola 2023	6760	67	5663326	15.3
Targowe 2022	4940	87	4585180	10.7

Table 2: Voter's and project's attributes space.

Voter's Attributes	Project's Attributes
Age	Cost
Gender	Category
Voting Method	Population segments

5 EVALUATION

In order to evaluate our predictions methods we divided our dataset into 3 sets: Train-set, Validation-set and Test set.

In our research context, the train-set is the data that we collected from voters regarding their votes. The validation set is predefined set of votes from closed set of voters, these votes are used to tune the hyperparameters of a model. We denoted that the validation set size is 15% of each data set. The test-set is the collection of votes that we use to evaluate our model, these votes are the votes that we did not collected from the voters. As mentioned in Table 4, the PB datasets are imbalanced, this can lead to biased results and sub optimal performance of our predicting models, as the model is more likely to predict the majority class. We address this issue by modifying the model loss function to give more weight to minority class samples.

We divide our evaluation methods into two sections: the *classification accuracy metrics* in which analyze the full predicted ballots and *Bundle evaluation metrics* that analyze the final predicted bundle.

5.1 Classification Accuracy Metrics

To analyze our algorithms, we considered a variety of *classification accuracy metrics*, namely: precision, recall and f1 [20, 23]. They measure the amount of correct and incorrect classification and are derived from *confusion matrix*. The *confusion matrix* holds the following measures: The acronym TP, FN, FP, and TN of the *confusion matrix* cells refers to the following: TP = true positive, the number of positive cases that are correctly identified as positive, FN = false negative, the number of positive cases that are misclassified as negative cases, FP = false positive, the number of negative cases that are incorrectly identified as positive cases, TN = true

negative, the number of negative cases that are correctly identified as negative cases [20, 23]. *Precision*, defined as $TP/(TP + FP)$, in RS context is the ratio of the number of relevant recommended items to the total number of recommended items [20, 23]. *Recall*, defined as $TP/(TP + FN)$, is the ratio of relevant recommended items to the number of relevant items.These two metrics, share an inverse relationship between them. Precision and recall are not sensitive to changes in data distributions. A perfect model will capture all positive examples (Recall = 1), and score as only the examples that are in fact (Precision = 1), from an analytical point of view it is desirable to increase recall without sacrificing accuracy. F_1 measure combines recall and precision as harmonic mean of them, $F_1 = (2 * precision * recall)/(precision + recall)$ and is suitable measure for an imbalanced data [23].

5.2 Bundle evaluation metrics

Eventually, we have interest in computing the bundle of winning projects that was decided by the voters, i.e, the list of projects that 'won' and would be implemented. As an evaluation method we compared the bundle from the *'real'* votes to the bundle of the *'predicted'* votes. This comparison has done by computing the *Symmetric Distance* between the the real bundle (rb) and the predicted bundle (pb).

5.2.1 Symmetric Distance. Symmetric distance(SD) is a distance measure that satisfies the property of symmetry, which states that the distance between two points is the same in both directions. Formally, given two points x and y, a SD function f satisfies the equation f(x,y) = f(y,x). Consider several toy examples: The SD between $rb = \{1, 2, 3\}$ and $pb = \{2, 1, 3\}$ is 0. The SD between $rb = \{1, 2, 3\}$ and $pb = \{1, 3, 4\}$ is 1. The SD between $rb = \{1, 2, 3\}$ and $pb = \{1, 4, 5\}$ is 4.

5.2.2 Fractional Allocation score. Another metric we use to evaluate the bundle is 'Fractional Allocation' (FA) Score ; We denote FA score as the sum costs of projects that were predicted properly divided by the cost limit - budget.

Definition 5.1 (Fractional Allocation (FA) Score). We define the *fractional allocation score* to be: $FA = \frac{\lambda}{B}, \lambda = \sum_{p \in pb \cap rb} cost(p)$

6 EXPERIMENTS

In this section we describe the experiments we conducted on the real-world data-sets presented in Table 1. We began with defining the partiality levels of the experiment, we defined multiple levels in order to conduct several experiments with different Sample degree and LV degree as mentioned in Section 3.0.1. The range of sample degrees is 0.1, 0.15, 0.3, 0.5, 0.7 and 0.9 and the range of LV degrees is 0.1, 0.2, 0.3, 0.5, 0.7, 0.9 and 1. When LV degree is equal to 1 it represent the case of sample size of the specific Sample Degree. After we established the frame of the amounts of information we would collect from the voters, we used each of our sample modules (Random, Offline-popularity, Offline-controversial, Offline-consensus and Online) to gather the preferences from voters. We performed each sample module on all sample degree-LV degree combination for 20 times. This procedure was executed on all PB datasets.

Then we performed each prediction module on the each of family of sample degree and LV degree. We did this procedure on every dataset as detailed in Figure 4 for 50 times. We assume that the more data we collected, higher sample degree in our context, the closer we will be to the true real ballot results. In particular, we assume that the FA score would increase as the sample degree increase and the SD would decrease the sample degree increase for each setup for every PB. Hence, we conducted a sanity test to check this assumption.

Sampling Module	Prediction Module	Sample Degree	LV Degree
Random	{Classification, MF, FM }	{0.1, 0.15, 0.3, 0.5, 0.7, 0.9}	{0.1, 0.15, 0.3, 0.5, 0.7, 0.9, 1}
Offline-popularity			
Offline-controversity			
Offline-consensiuos			
Online			

Figure 4: Treatment Matrix - please see Section 6.

Every algorithmic setup holds different level of intervention in selecting the project proposals to collect; in the 'Random' setup the algorithm has no power at all in choosing which project proposals preferences to collect, in the offline setups it has the power to choose K project proposals based on group of voters that provided full ballots (LV group) and for the online setup it has the power to choose project proposals based on all voters preferences itteratively. We assume that the more 'power' the algorithmic solution hold the better it would perform better (higher FA score and lower SD).

Hence, for every PB we compared all setups and algorithmic solutions per sample degree. Additionally, we implemented a performance test in which we compared our solutions with a naive sampling procedure. This sampling procedure is the case where LV degree is equal to 1 for a certain sample degree. Note that in sampling procedure only part of community (sample degree to be exact) share their preferences, in contrast to our proposed solutions where every voter has at least K project proposals to share her opinion on. In order to generate it we randomly sampled voters for each PB for each sample degree for 50 times and calculate its FA and SD.

7 RESULTS

In this section we describe and discuss our results for the prediction module, for the sampling module, and for the architecture as a whole.

First we performed a sanity check to test how the sample and LV degrees effect on FA scores and SD. We calculated the average FA scores and SD for all PB's for each combination of sample degree and LV degree as described in Figure 4. The sanity test results, shown in Figures 5 and 6, showed that in most LV degrees, as the sample degree increases the FA score increases as well and the SD is decreases, as assumed.

Furthermore, it can be observed that as the LV degree increases the FA score/SD increases/decreases as well, by means the prediction modules perform better when a larger group of voters provide their full ballots.

In Figures 7 - 12 we can see the FA scores performance of every setup as function of LV degree and sample degree including a comparison to sampling method (LV degree equals to 1). It seems that in some settings some of our solutions succeed better than a sampling

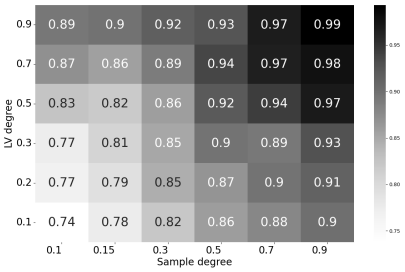


Figure 5: Sanity test - heatmap of FA scores as function of Sample degree and LV degree.

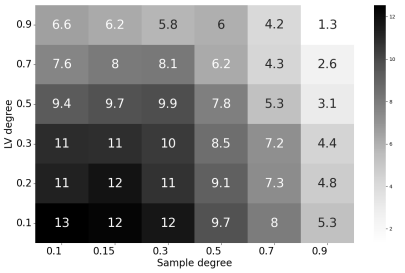


Figure 6: Sanity test - heatmap of SD as function of Sample degree and LV degree.

method. Specifically, for sample degree equals to 0.1 (Figure 7) all solutions except MF-online, MF-offline-popularity and FM-online produce higher FA scores than sampling method. Specifically the Classification-online and Classification-off-popularity setups for sample deg equals to 0.1 perform better than sampling method for sample degree equals to 0.3. In Figure 8 it seems that our solutions perform not as good as the sampling method, except from Classification-online and Classification-off-popularity. In Figure 9 we can see that more of our solutions perform better or good as good as the sampling method, except from Classification-online and Classification-off-popularity. In Figure 10 we see that the random setups, for all prediction modules, outperforms and achieved relatively high FA scores. Additionally, the Classification-off-popularity gained the maximal FA score, we achieved it for collecting half of the votes. As for sample degree equals to 0.7 and 0.9 it is not surprising that our solutions gained high FA scores. Through all sample degrees the setups that perform best, produce the higher FA scores, are the Classification-online and Classification-offline-popularity setups, especially in the 0.1, 0.15 and 0.3 sample degrees. For higher sampling degrees all of our solutions perform better or good as sampling methods. As we assumed, the random sampling module perform worse than the offline and online sampling modules. As for the prediction modules, it is shown that the classification technique performs better than the RS techniques.

In Figures 13 - 18 represent the SD of every setup as function of LV degree and sample degree including a comparison to sampling method. We see quite the same trend as we saw for for the FA scores, Classification-online and Classification-offline-popularity setups perform best for all sample degrees.

8 CONCLUSIONS AND OUTLOOK

We have proposed the use of techniques from machine learning and recommendation systems to tackle the information overload problem in participatory budgeting. Technically, we have developed a more of PB with partial ballots that is useful for our context, designed several algorithmic solutions for different specific prediction settings, and reported on the results of computer-based simulations showing the high quality of prediction with respect to voter ballots and the winning bundles that the algorithms achieve for a real-world data. We conducted our experiments on 10 real-world PB and eliminated some preferences to simulate PB with partial ballots. This elimination of preferences were conducted under controlled and structured manner, defined as sampling module. After we created these partial ballots we implemented prediction module to predict the missing preferences. Then we calculated the winning bundle using greedy approval voting rule. We evaluate our sampling and prediction modules by comparing the winning bundles to the real ones. The comparison were defined by symmetric distance and Fractional Allocation score, as a measure of how good the budget was allocated in the predicted winning ballot. Except from comparing our own solutions between themselves, we compared them to a naive sampling method. We found that some of our solutions perform better from sampling method. Hence, although taking a sample of PB community and extract winning bundle out of it's preferences is quite easy and performs relatively good, we propose solution that take into account all of PB's voters and performs better than sampling methods. So, our solutions permit all potential voters in a PB to express their opinions regarding project proposals and provide a solution to the information overload problem.

Some avenues for future research are the following:

- A different, somewhat stronger setting may correspond to algorithms that initially get an instance of PB with partial ballots where **all** voter preferences are hidden (i.e., where $TV = V$ and $E = \emptyset$ for all $v \in TV$). Designing and analyzing algorithms for this setting is of practical importance, as, inherently, a PB process starts with only completely empty ballots. From a technical point of view, this would naturally allow the algorithm consider more complex correlations across the electorate.
- A more interactive setting that is worth considering may be the following: first, the voter reveals some of her preference, after which the algorithm estimates the rest of the ballot (or, alternatively, only the preferences of the voter regarding some further projects not yet revealed). Then, the voter has the possibility to examine the predictions done by the algorithm and to "correct" some of them, in which case the algorithm can further update its estimates regarding the remaining projects. Such an interactive communication between a voter and the envision application may lead to results of higher quality.
- While here we have considered the quite standard PB setting of approval-based combinatorial PB, analyzing a similar RS-oriented approach for different settings of PB (in particular,

those in the survey of Aziz and Shah [2]) is a natural avenue for future research.
- While we proposed certain approaches to address the prediction module, it would be interesting to imply other machine learning techniques to predict voter's ballot.

REFERENCES

[1] Haris Aziz, Barton Lee, and Nimrod Talmon. 2017. Proportionally representative participatory budgeting: Axioms and algorithms. *arXiv preprint arXiv:1711.08226* (2017).

[2] Haris Aziz and Nisarg Shah. 2021. Participatory budgeting: Models and approaches. In *Pathways Between Social Science and Computational Social Science*. Springer, 215–236.

[3] Gerdus Benade, Swaprava Nath, Ariel D Procaccia, and Nisarg Shah. 2021. Preference elicitation for participatory budgeting. *Management Science* 67, 5 (2021), 2813–2827.

[4] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* 46 (2013), 109–132.

[5] Felix Brandt, Vincent Conitzer, and Ulle Endriss. 2012. Computational social choice. *Multiagent systems* 2 (2012), 213–284.

[6] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.

[7] Yves Cabannes. 2004. Participatory budgeting: a significant contribution to participatory democracy. *Environment and urbanization* 16, 1 (2004), 27–46.

[8] Iván Cantador, María E Cortés-Cediel, Miriam Fernández, and Harith Alani. 2018. What's going on in my city? recommender systems and electronic participatory budgeting. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 219–223.

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

[10] Nimrod Talmon Dariusz Stolicki, Stanisław Szufa. 2020. pabulib an open PArticipatory BUdgeting LIBrary. http://http://pabulib.org.

[11] Angela Edmunds and Anne Morris. 2000. The problem of information overload in business organisations: a review of the literature. *International journal of information management* 20, 1 (2000), 17–28.

[12] Peter Emerson. 2013. The original Borda count and partial voting. *Social Choice and Welfare* 40, 2 (2013), 353–358.

[13] Wulf Gaertner. 2009. *A primer in social choice theory: Revised edition.* Oxford University Press.

[14] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. 2010. *Recommender systems: an introduction.* Cambridge University Press.

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[16] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. 2014. Facing the cold start problem in recommender systems. *Expert Systems with Applications* 41, 4 (2014), 2065–2073.

[17] Ian MD Little. 1952. Social choice and individual values. *Journal of Political Economy* 60, 5 (1952), 422–432.

[18] Tom M Mitchell and Tom M Mitchell. 1997. *Machine learning*. Vol. 1. McGraw-hill New York.

[19] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[20] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.

[21] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 291–324.

[22] J Ben Schafer, Joseph Konstan, and John Riedl. 1999. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*. 158–166.

[23] Gunnar Schröder, Maik Thiele, and Wolfgang Lehner. 2011. Setting goals and choosing metrics for recommender system evaluations. In *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*, Vol. 23. 53.

[24] Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender systems handbook*. Springer, 257–297.

[25] Nimrod Talmon and Piotr Faliszewski. 2019. A framework for approval-based budgeting methods. In *Proceedings of AAAI '19*, Vol. 33. 2181–2188.

[26] Chen Wang, Chengyuan Deng, and Suzhen Wang. 2020. Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. *Pattern Recognition Letters* 136 (2020), 190–197.

[27] Yueping Zheng and Hindy Lauer Schachter. 2017. Explaining citizens' E-participation use: The role of perceived advantages. *Public Organization Review* 17, 3 (2017), 409–428.

APPENDIX

We provide some further plots in the appendix. Figures 7 - 12 describe the FA scores that were produced for every combination of the algorithmic solutions, prediction module, LV degree and sample degree as describe in Figure 4. These heatmaps enable us to compare the FA scores between the algorithmic solutions (random, offline-popularity, offline-consensus, offline controversy and online) and between the prediction module (MF, FM and binary classification). Furthermore, we compare between our solutions across all LV degree levels to sampling solution, where only part of the voters taken into account, it is the case where LV degree equals to 1, where each figure refers to different sample degree.

Figures 13 - 18 describe the same comparisons shown above but for Symmetric distances (SD).

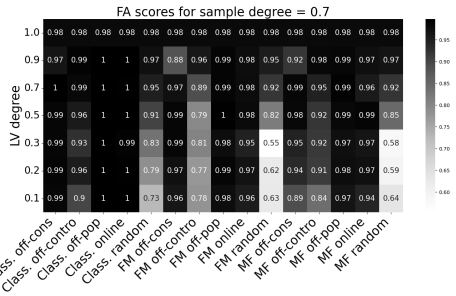


Figure 11: Heatmap of FA for Sample degree = 0.7.

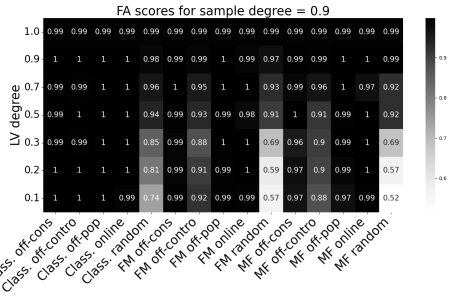


Figure 12: Heatmap of FA for Sample degree = 0.9.

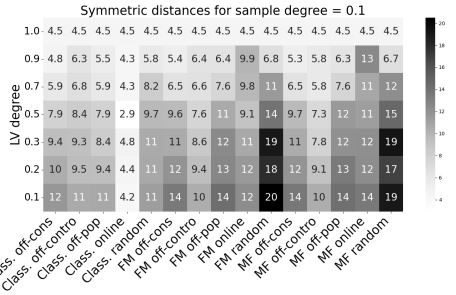


Figure 13: Heatmap of SD for Sample degree = 0.1.

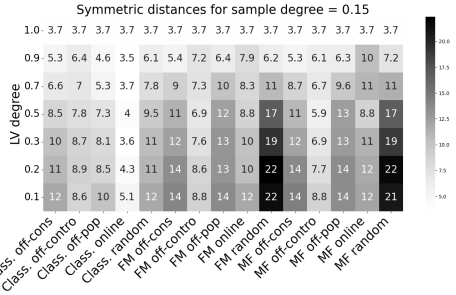


Figure 14: Heatmap of SD for Sample degree = 0.15.

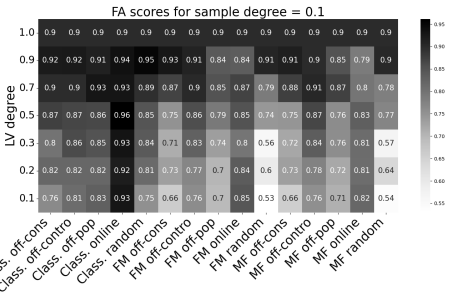


Figure 7: Heatmap of FA for Sample degree = 0.1.

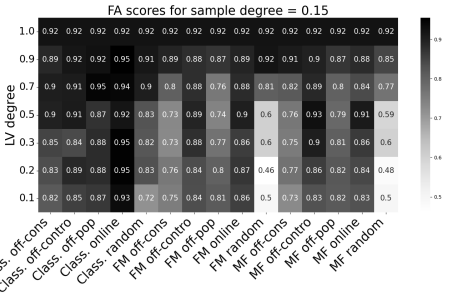


Figure 8: Heatmap of FA for Sample degree = 0.15.

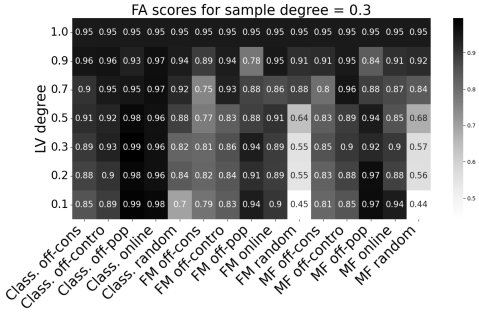


Figure 9: Heatmap of FA for Sample degree = 0.3.

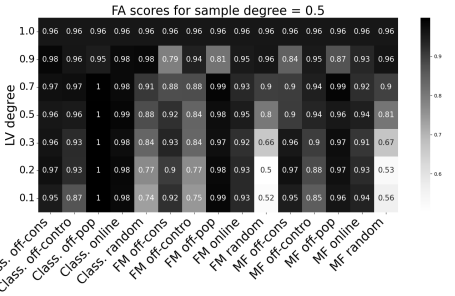


Figure 10: Heatmap of FA for Sample degree = 0.5.

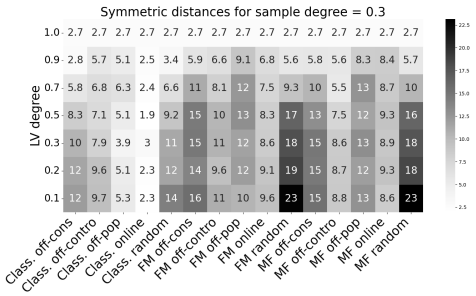


Figure 15: Heatmap of SD for Sample degree = 0.3.

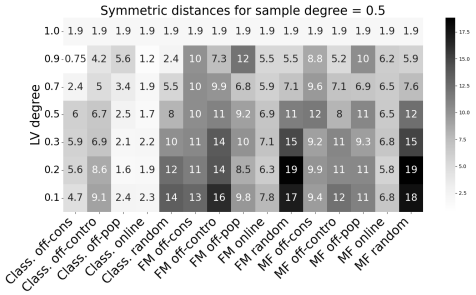


Figure 16: Heatmap of SD for Sample degree = 0.5.

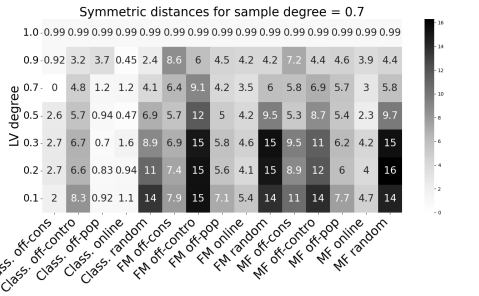


Figure 17: Heatmap of SD for Sample degree = 0.7.

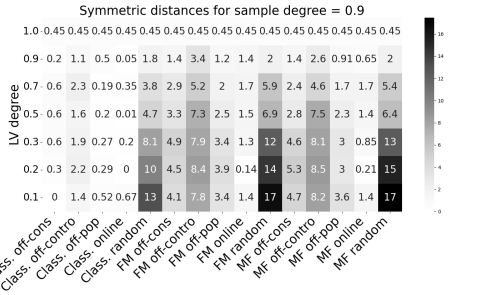


Figure 18: Heatmap of SD for Sample degree = 0.9.