Improve the performance-compatibility tradeoff using individualized objective functions in the context of human-agent interaction.

Authors: Shalom Rosh, Alexander Shachor and Avraham Miletzky. Undergraduate students in the Department of Computer Science at Bar Ilan University

Abstract:

J. Martinez et al. (2021), the authors of *Improving the Performance-Compatibility Tradeoff with Personalized Objective Functions* describe a method to improve the performance compatibility tradeoff. By using this approach, we seek to improve the human-agent interaction and improve user confidence in the decision-making assistance system. The code, models, and datasets will also be discussed, as well as the results and conclusions.

Introduction:

Artificial intelligence is here to stay. Commercial companies and business organizations are increasingly using AI and related technologies. Some AI systems assist people by providing recommendations for books, movies, and songs based on personal preferences, as well as decision-making systems, such as medical AI systems.

In response to the user's interaction with the AI system, two processes occur. Initially, the user formulates some expectations about the capabilities of the system based on the quality of its recommendations. In addition, the system collects more data and can update its model prediction. As much as updating the model can improve the system's performance, it can also alter the system's way of making predictions, which the user may not expect based on past interactions. As a result, although an update can improve the overall performance, it may not be compatible with the user's expectations, causing the user to lose trust and disregard the recommendations. Alternately, follow the wrong recommendations that the model, before the update, got right.

A simple example will illustrate the issue. Look at a doctor who uses an AI-based model to determine whether a mole on the face is cancerous. Assume the current model is 70% accurate. As the dataset grows, the model is updated and has an 85% accuracy rate. However, this new model has deteriorated in terms of detection moles on the face, achieving an accuracy of only 60%. Ideally, the doctor will discover that there is a problem and stop trusting the system. At worst, the doctor will continue to rely on the system, resulting in disastrous results.

A naive approach is to force compatibility with the previous model. However, this reduces the general performance of the new model and limits the use of the new data. Martinez et al. suggest a method of altering objective functions so that high performance is maintained without compromising user trust.

In human-agent interaction, all aspects of human-machine interaction are taken into consideration, including its improvement and its reliability. Thus, solving the compatibility-performance tradeoff problem will improve human-agent interactions, since improvements for one will not reduce compatibility-performance for another, and enhance user confidence in decision-making assistance systems. Furthermore, implementing Martinez et al. solution of personalizing the objective function assures a good human-agent interaction, as the compatibility for each user remains high.

Martinez et al. methodology:

Martinez et al. propose a personalized approach that increases user trust in the AI system and improves the compatibility-performance tradeoff. They proposed to customize the loss function for each user, which requires several additional settings. Let h_1 and h_2 represent the pre-update model and the post-update model, respectively. Such that h_1 is trained on a small subset of the dataset D that is used for h_2 training. Hypothesis h is correct for dataset D if $\forall\, x_i \in D$ we have that $h(x_i) = y_i$. $D^c \subseteq D$ is a subset of samples for which h_1 is correct. C stands for **C**ompatibility since errors that h_2 makes on samples in D^c decrease its compatibility score. Let $D_i \subseteq D$ be a subset of samples related to the history of interaction between user i and the AI-system, and $D_i^c \subseteq D_i$ be the subset of those samples such that the pre-update hypothesis h_1 is correct. Errors that the updated model h_2 makes on D_i^c are newly introduced errors that reduce their compatibility score.

Martinez et al. personalize the objective function to a target user i as follows: The objective function gets the user i and vector W- a four-dimensional weights vector for the defined groups D, D_i, D^c and D_i^c. Each weight indicates the effect of that group on the new model for the specific user, the L_m function defined as follows:

$$L_m(x, i, D, W, \lambda) = (1-\lambda)\cdot(w_0 \cdot \mathbb{1}[x \in D] + w_1 \cdot \mathbb{1}[x \in D_i])\cdot L(x) + \lambda \cdot (w_2 \cdot \mathbb{1}[x \in D^c] + w_3 \cdot \mathbb{1}[x \in D_i^c])\cdot L(x)$$

Where $\mathbb{1}$ is the indicator function, and $\lambda \in [0,1]$ is the importance given to compatibility. For a particular case x, $L(x)$ is the penalty (loss) given to an updated model for the label it predicted. As the λ parameter increases, the penalty for new errors increases as well as the compatibility, resulting in a new model more compatible with the old model.

If sample x belongs to user i's history, then x belongs to D and D_i. If the pre-update model $h_1$predicts the correct label for this sample, then x belongs to D^c and D_i^c. By assigning the sample weights in this manner, Martinez et al. personalize the objective function concerning the target user i according to the personalization approach indicated by W. By varying the weight parameters in W we can generate models that differ in how much the objective function is personalized. As shown in **Table 1,** Martinez et al. considered only binary weights and nine models - M_i were obtained. Note, there are 9 models rather than 16 because all the other models

model	W_k			
	w_0	w_1	w_2	w_3
m_0	1	0	1	0
m_1	0	1	0	1
m_2	0	1	1	0
m_3	0	1	1	1
m_4	1	0	0	1
m_5	1	1	0	1
m_6	1	0	1	1
m_7	1	1	1	0
m_8	1	1	1	1

Table 1: The portfolio of models M.

do not matter, since if w_2 and w_3 are 0, compatibility is not considered. Also, m_0 vector is called the baseline model, as it does not distinguish between users.

After updating the weight function, selecting a personalized model m_k for a target user i is required to compute the weight of each sample. Martinez et al. propose using a nested cross-validation process. This cross-validation process, described in **Figure 1**, preserves the temporal consistency of the user's interactions with the system and consists of two levels: An outer cross-validation loop for improving the statistical significance of the results and an inner cross-validation loop for selecting the best model for each user.

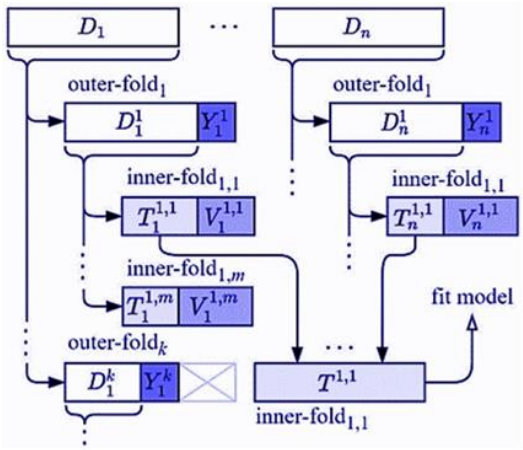


Figure 1: Visualization of the nested cross-validation with k outer-folds, m inner-folds and n users. D_i is the data of user i. In each outer-fold$_p$ a test set Y_i^p is selected, while in each inner fold$_{p,q}$ the remaining data D_i^p is split into a train set $T_i^{p,q}$ and a validation set $V_i^{p,q}$. The models are fitted on $T^{p,q}$ (the union of all the user train sets). The crossed boxes indicate the data dropped as part of the cross-validation process.

Let D_i be the time-series data available for user i in dataset D. In each outer-fold$_p$ a time-contiguous test set $Y_i^p \in D_i$ is chosen for each user i. Let D_i^p be the set of data points representing user i's interactions that happened before all interactions in Y_i^p. We use D_i^p for the inner cross-validation loop. In each inner-fold$_{p,q}$ a timestamp is randomly selected for every user i and used to split D_i^p into a disjoint and time-contiguous training set $T_i^{p,q}$ and validation set $V_i^{p,q}$. We choose the timestamp defining the split randomly to vary the amount of training data for each user. Every model $m_k \in M$ is trained on $T^{p,q}$and validated on $V_i^{p,q}$. Each sample $x \in T^{p,q}$ is assigned the weight $w(x, i, T^{p,q}, W_k, \lambda)$ (L_m with the respective weight vector W_k from Table 1). The model with the best average performance on the sets $V_i^{p,q}$, where we vary q and fix both i and p, is chosen for the target user i. Finally, the chosen model fitted on D_i^p, for maximizing the amount of training data, and evaluate on the test set Y_i^p. For user i, each iteration of the outer loop provides a prediction and thus a quality metric of the system. As this method yields different results for different users, an average is necessary to compare its results to the single result obtained from the baseline model.

By plotting each point for a different value (**Figure 2**), regarding the performance-compatibility tradeoff, model quality can be analyzed. It is like the ROC graph, which illustrates how well a classification model's predictions are matched according to its true-positive and false-positive ratios. AUC provides a measure of comparison between different models by selecting the model with the largest area under the curve. In the same way, models

can be compared according to their performance-compatibility tradeoffs by measuring the AUTC (Area Under the Tradeoff Curve).

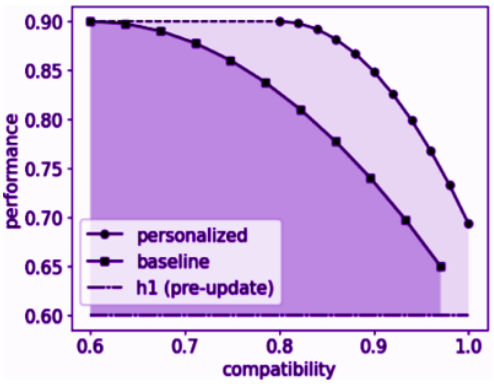


Figure 2 : Like ROC metric, we can measure the quality of a performance-compatibility tradeoff by computing the area under its curve.

In **Figure 2,** the x-axis represents the model's compatibility score, and the y-axis its performance (e.g., accuracy). The dashed horizontal line corresponds to the performance of the pre-updated model h_1, while the two curves represent two updated models fitted using different W vectors. Each point corresponds to a particular value of λ, the compatibility parameter. The personal model of Martinez et al. achieved a better result than the naive model, as it showed that the AUTC of the personal model was larger than the standard model. A dashed line completes the curve of the personal model, which begins at a higher compatibility value due to its good performance, and without completing it, the area under the curve would be small and would not correctly illustrate the quality of the model.

Methods And Materials:

In order to test the method, the authors applied it to domains that are commonly used in Human-Computer Decision Making. The chosen domains are following the next criteria; One, include multiple interactions between users and the system. In addition, the system makes decisions in real-time. Finally, the system could be used to assist in user decision-making.

To conduct the empirical experiment, three domains were selected according to the above-mentioned criteria:

ASSISTments - This problem includes interactions of students who solve online math problems in classrooms. **The classification task** - Determine whether a student will answer a question correctly on the first attempt, based on his previous experience with the system. **The Dataset** - In this case, the dataset size was approximately 50,000 samples for about 100 students. Each student had between 50 to 1000 pieces of information, with an average of 525.

CitizenScience (CS) - This is a citizen science project named Galaxy Zoo, which invites people to assist in the morphological classification of galaxies from images. **The classification task** - determine if a user will drop out from the GZ site within a designated time window. **The Dataset** - The dataset included ~10000 samples for 20 users. For each user, there were between 100 to 1900 samples, with an average of 1035.

MOOCPosts (MOOC) - This is Stanford University's dataset that contains student posts from eleven-course forums. **The classification task** - Determine whether students are confused about the course material based on their posts. **The Dataset** - It was a dataset of 5,000

samples for approximately 100 students. For each student, there were between 20 to 500 pieces of information, and on average, there were 260.

The three domains were tested via code. The code uses 3 classifiers, Decision Tree Classifier, Random Forest Classifier, Xgboost Classifier, and applies them to each domain. In the code, data is divided according to users, and the outer loop makes sure that the results are statistically accurate, whereas the internal loop is used to choose the best model. Thus, in the internal loop, each sample is given a weight that corresponds to the model being studied by initializing the weight parameter of the chosen classifier based on the weight of the model (e.g., for the model m1 weight parameter will be 1,0,1,0). To generate the AUTC, each model trains with different λ values.

In this section, we will describe each module of the code for the project:

Runner.py - This module represents the starting point of the code; it contains the main function and some other classes and functions responsible for init time and saving results to log files.

NestedCrossValidation.py - This module is called by the module Runner.py and it is responsible to create and perform the nested cross-validation process on the chosen dataset.

ExperimentSettings.py - This module allows you to specify how the experiment should be configured, including which data set should be analyzed with the specified parameters, as well as the files physical locations.

DataPreparation.py - This module takes care of preparing the preexisting data for the AI algorithm using the calculated and prepared parameters. Also, creating the files and folders.

Models.py - This module provides parameters for evaluation and modeling functionality, including model fitting, model prediction, score, etc. Initialization includes the desired model as well as optional parameters.

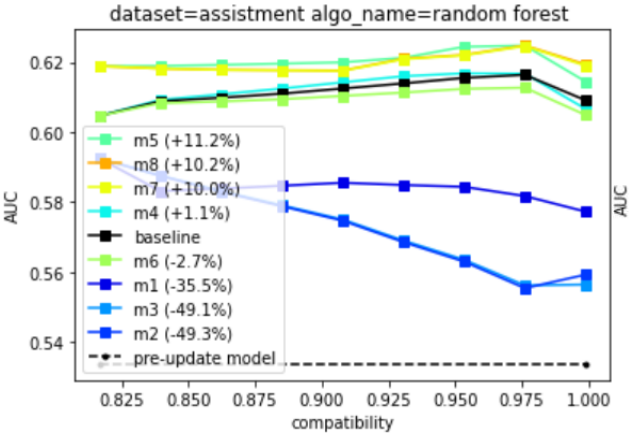
AnalyseResults.py - This module is responsible for processing the results, analyzing and presenting them visually to compare and examine the personal approach in relation to the baseline solution.

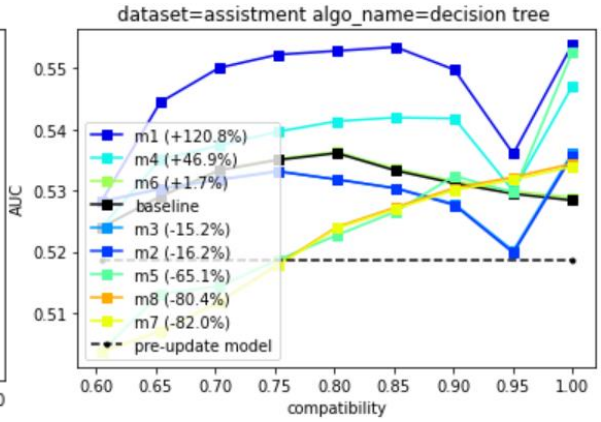
Results And Discussion:

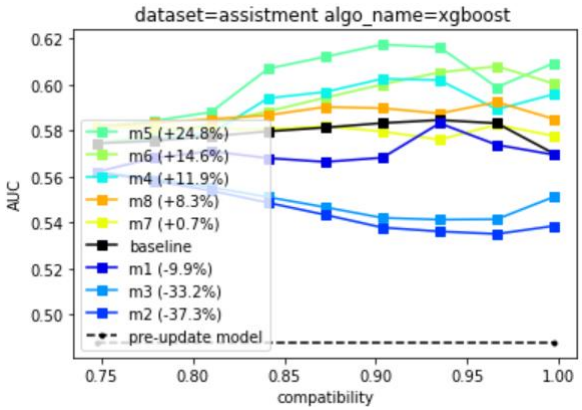
Unlike Martinez et al, we tested the method with Random Forest, Xgboost as well as Decision Tree. Our decision to use a Random Forest is based on the following reasons: One, Random Forests are considered a highly accurate and robust method due to the number of decision trees participating in the process. It does not suffer from the overfitting problem. It averages all predictions, reducing the biases. Moreover, as for Random Forests, they combine predictions from a large number of decision trees. The logic is that even a single model made up of many mediocre ones will still be better than a single great one.

We decide also to use Xgboost, since Random Forest is what we call "bagging applied to decision trees" (reduce variance through bagging), which keeps the model relatively stable even if the data changes. Xgboost based on **Boosting** which reduces variance and bias. The use of multiple models (bagging), reduces variance. It reduces bias too by training the subsequent model by the information of what errors the previous models made (the boosting part).

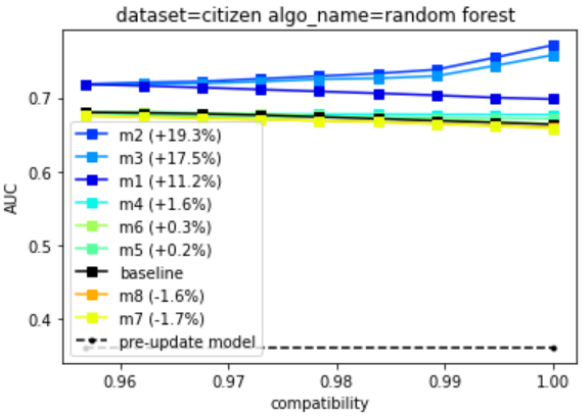
ASSISTments dataset gave the following results:

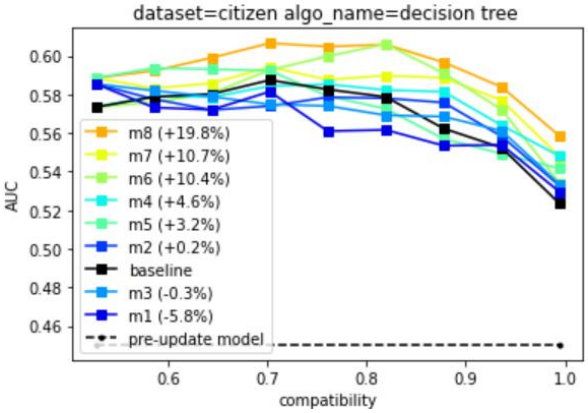


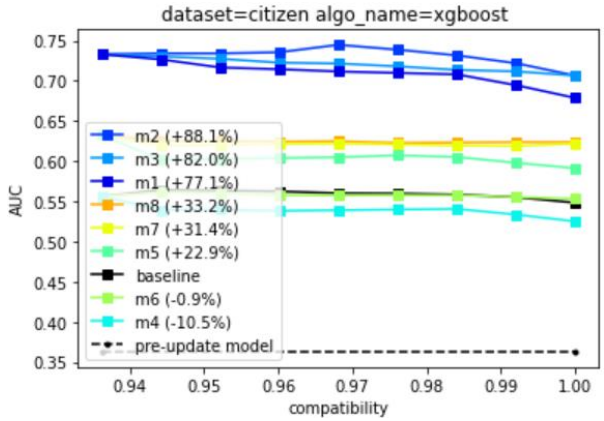




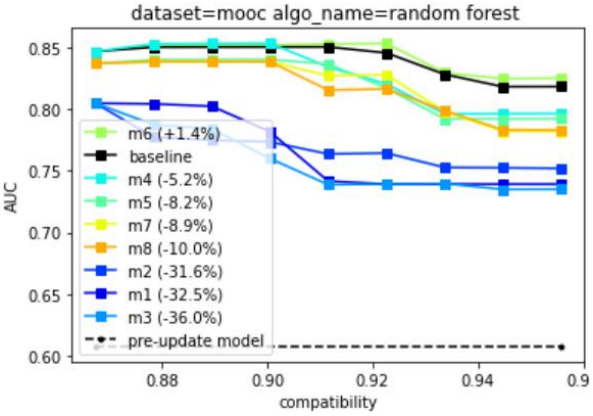
The **CitizenScience** dataset yielded the following results:

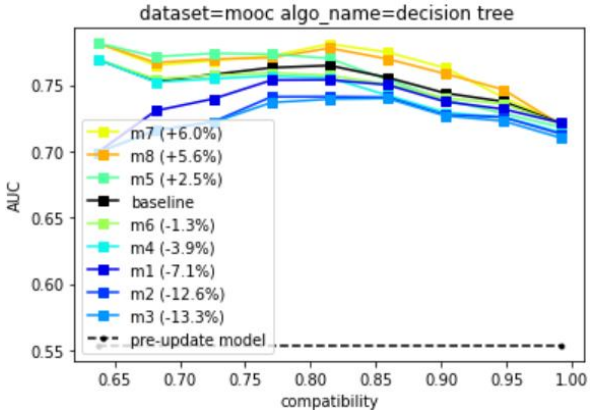


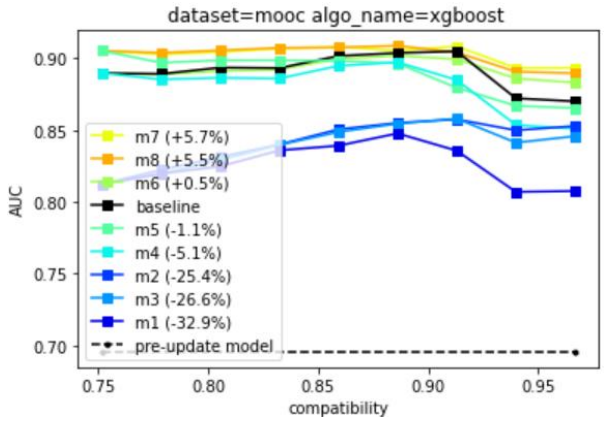




MOOCPosts dataset provided the following results:







As observed on these datasets, the results indicated that the Martinez et al. method consistently provides better results than the baseline model. Moreover, as can be seen from the results of the CitizenScience and MOOCPosts datasets, the results are significantly improved (the area below the AUTC curve increases) when using more complex models like Random Forest and Xgboost for these datasets, which further improves the Martinez et al. method of improving performance-compatibility tradeoff using personalized objective functions.

Conclusions:

In this article, we explain and test the Martinez et al. method related to the performance-compatibility tradeoff. Based on the results of our empirical experiments, the method certainly contributes to the improvement of the performance-compatibility tradeoff. Furthermore, regarding the Martinez et al. method, we demonstrated that other machine learning models like Random Forest and Xgboost can improve the method used by Martinez et al. and achieve better results.

A better performance-compatibility trade-off, as the results above, benefits users on both aspects. As a group, the users benefit from an AI system that performs better and thus, makes more accurate and reliable suggestions. As an individual, each user enjoys the compatibility of the suggestions which he used to get. Therefore, the users continue to trust and use the AI system, which improves model predictions and so on.

Further improvement that might be done - Regarding Martinez et al. approach to choose models only composed of {0,1}, is worth examining. In some cases, it may be beneficial to group models from the [0, 1] range. But there are two potential downsides doing so. To begin with, we'll have a lot of models, which will influence running time. Secondly, we might not know which model to select, since the inner loops may produce different results. This scenario is more likely to occur since the models are close together and there are many of them. Despite its flaws, this theory is still worth exploring and testing.

We suggest another improvement, rather than treating each user individually, data can be grouped by several parameters, such as age, gender, height, and any other parameters that are relevant to the specific service that the machine learning model trained on. Users can be divided into coherent groups, and on these groups, we can apply the authors' methodology. Using this technique improves the heavy runtime by reducing the number of runs since each user is not trained individually, rather than with a group.

A significant advantage of dividing by groups is when it comes to data sets that represent distinct groups of users, where these users share similarities, and these similarities can help identify them as a group. The iteration over groups, in this case, improves both the results for all the users, as they are part of the groups, and the overall runtime. Users without much data ("cold start") benefit from the results of similar users with more data since they are in the same group and have similar behavior. A variety of clustering algorithms are available to identify different groups from a data set when the grouping is unclear, and then the above method can be used. In our view, the above suggestions should improve the model performance as well as diminish the runtime obstacle, but more empirical studies are required to confirm this.