CACTUS: Detecting and Resolving Conflicts in Objective Functions

Subhajit Das and Alex Endert

Abstract—Machine learning (ML) models are constructed by expert ML practitioners using various coding languages, in which they tune and select model hyperparameters and learning algorithms for a given problem domain. They also carefully design an objective function or loss function (often with multiple objectives) that captures the desired output for a given ML task such as classification, regression, and others. In multi-objective optimization, conflicting objectives and constraints is a major area of concern. In such problems, several competing objectives are seen for which no single optimal solution is found that satisfies all desired objectives simultaneously. In the past, visual analytic (VA) systems have allowed users to interactively construct objective functions for a classifier. In this paper, we extend this line of work by prototyping a technique to visualize multi-objective objective functions either defined in a Jupyter notebook or defined using an interactive visual interface to help users to detect and resolve conflicting objectives. Visualization of the objective function enlightens potentially conflicting objectives that obstructs selecting correct solution(s) for the desired ML task or goal. We also present an enumeration of potential conflicts in objective specification in multi-objective objective functions for classifier selection. Furthermore, we demonstrate our approach in a VA system that helps users in specifying meaningful objective functions to a classifier by detecting and resolving conflicting objectives. Through a within-subject quantitative and qualitative user study, we present results showing that our technique helps users interactively specify meaningful objective functions by resolving potential conflicts for a classification task.

Index Terms—Objective functions, Conflict resolution, Classification, Machine Learning, Human-in-the-loop

1 Introduction

Traditionally machine learning (ML) experts construct models by writing code for finding the right combination of hyperparameters and learning algorithms, and specifying an appropriate objective function (also called loss/cost functions) to the modeling task. In the past, researchers in visual analytics (VA) have investigated making ML model construction interactive, which means developing visual interfaces that allow users to construct ML models by interacting with graphical widgets or data marks [1], [2], [3], [4], [5]. Recently, Das et al. introduced a VA system, QUESTO [6], that facilitates interactive creation of objective functions to solve a classification task utilising an Auto-ML system. While their approach helped users to interactively explore and express a wide array of objectives to an objective function, the authors discussed potential conflicts that may occur in interactive specification of objectives as a limitation to QUESTO's workflow.

Interactive specification of objectives and constraints may result in objective functions with conflicting objectives [7], [8]. Conflicting objectives may cause construction of inefficient objective functions that may confuse the underlying algorithm (e.g., an Auto-ML model solver) due to unclear user goals. For example, in a classification task, a user may expect to see similar data items in the same class label; at the same time they may also expect that the global accuracy of the model is high for every class. The model being trained to support the users request to place similar data items in the same label category, may not perform equally well for all class labels, thus dropping the global accuracy of the model. In the past conflicts in objective specification in multi-objective objective functions was addressed using trade-off analysis [8], [9], [10], [11]. While useful, to our knowledge there is no other work in VA that have looked at detecting and resolving conflicts in user-specified objectives to build classifiers.

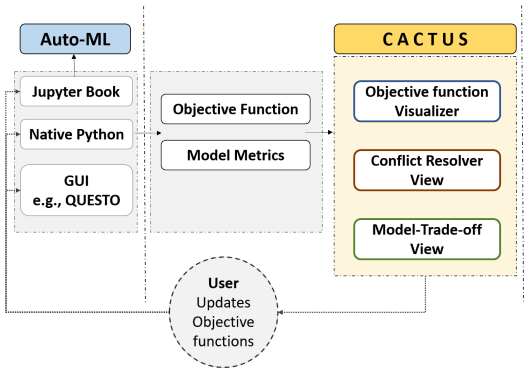

Fig. 1. CACTUS workflow for resolving conflicts in objective functions.

In this paper, we extend research on interactive objective functions by helping users detect conflicts in objective functions, and further interactively resolving these conflicts to specify a more meaningful objective function to model selection systems (e.g. Auto-ML). Grounded on extensive literature search, we enumerated a list of potential conflicts between various objectives in interactive objective functions explained later in the paper. Furthermore, we present a VA tool that facilitates: (1) Visualization of a multi-objective objective function that is defined using a Python code (e.g, defined in a Jupyter notebook), or a visual interface such as QUESTO [6], (2) Highlighting conflicts between interactively specified objectives, (3) Helping users to resolve these conflicts to improve their objective functions, and (4) Train multiple classifiers incrementally (over multiple iterations) to perform trade-off analysis of objectives in multi-objective objective function. As

• *Subhajit Das and Alex Endert are with Georgia Institute of Technology. E-mail: das, endert@gatech.edu*

objective functions drive all ML algorithms, we consider visualizing objective functions may help users (e.g., novice ML users, ML experts who may use GUI to debug models, etc.) to understand their specifications to the underlying models. Defining a meaningful and correct objective functions (free of conflicts) is very important to ensure models that are sampled from Auto-ML systems perform as per user expectations. With the proposed approach, in a Jupyter notebook one can define an objective function and then use our technique to visualize, test, and interactively adjust objectives to explore numerous model alternatives for their analytical ML task.

We prototyped CACTUS, a conflict resolution and trade-off analysis system for user specification of objectives. CACTUS ingests an objective function (defined in Python or Jupyter notebook) and then visualises it to show its objectives and respective weights. It visually explains the objectives that are closely satisfied by the selected classifier and the ones that failed to be satisfied (see Figure 1). Furthermore, it visualises conflicts between objectives, allowing users to incrementally improve the function by making adjustments interactively. When objective functions are adjusted interactively, users can re-train models, see a change in the models' performance and continue exploring the space of multiple variants of objective functions. Furthermore, visualising objective functions and the trained models' performance metric (per objective) can empower users to probe models in relation to how well they satisfy their goals. For example, users can probe if similar data items are predicted in the same class, or strong representative (candidate) data subsets are predicted with higher probabilities.

In addition, we quantitatively evaluated CACTUS to test if it helped users to find and resolve conflicts. We also present qualitative feedback from the participants that enlightens the strengths and weaknesses in CACTUS, its' potential usability issues, and limitations that needs further research. Our study showed that: (1) Participants found CACTUS intuitive, expressive and very effective in showing (and helping to resolve) conflicts between objectives. (2) CACTUS helped participants ideate on multiple versions of objective functions and in the process resolve conflicts in objectives. Our contributions are:

- An enumeration of potential conflicts in objective specification in multi-objective objective functions to construct classifiers.
- A VA system CACTUS that visualises conflicts in objective functions and supports interactive resolution of them.
- A within-subject quantitative and qualitative user study validating that our technique helps users construct meaningful objective functions by resolving conflicts between objectives.

2 Related Work

Interactive classification in VA: In machine learning, *classification* is a process to predict a class label of a data item (e.g., rows in a table) given its features in the data (e.g., columns in a table). Many VA systems made this process of classifier construction interactive [12], [13], [14], [15], [16], [17], [18]. In some cases, this interactive classifier construction process also supported data labeling [19], hyperparameter tuning [20], latent space visualization of complex models such as deep neural networks [21], etc. For example, a system called ModelTracker visualized model performance (e.g., validation set's accuracy) to debug and improve models by facilitating direct data inspection by ML practitioners [1]. These surveys further showed visualization systems that allowed construction of classifiers [22], [23]. People also researched interactive construction of multiple models e.g., the system Prospect allowed ML practitioners to interactively revise data properties. These models helped users understand the relationship between data features and the class label [24]. While these systems helped users (from novices to experts in ML) interactively build classifiers, in this paper we help people create many variants of objective functions by resolving conflicts between objectives as users intend to construct optimal classifiers.

User preferences in objective functions: We researched the literature to understand user specifications that goes into the construction of objective functions in ML [25], [26], [27], [28] and how these functions are visualized or made interactive in the past. User goals are often personalised and domain specific [29], [30]; users' may require custom-designed metrics to select models, as opposed to utilize known metrics such as 'accuracy' or 'recall' [31]. Often users may teach machines to show example data instances of a specific class label to help models learn how to discriminate between classes [25]. For instance, similarity between data instances or a subsegment of the data is utilized to express user expectation from classifiers [32]. Similarly, in the system Flock users specified features in the data instances that rendered them similar to be in a specific class label [33]. People in the past have visualized solutions recovered using multi-objective objective functions using charts such as RadViz, bubble chart, parallel coordinate plots etc. [34], [35], [36]. A 2D-polar coordinate plot was used to help users see trade-offs between objectives in this work [37]. Others have analysed variety in pareto-optimal solutions in multi-objective optimization functions [38], [39]. Das et al. prototyped QUESTO, allowing users to interactively create objective functions to select optimal classifiers trained on tabular data [6]. In this paper we seek to extend research on interactive objective functions by showing users, conflicts in objectives as opposed to allowing interactive creation of objective functions.

Conflicts in multi-objective objective functions: Real-world ML applications (e.g., finance, transportation, medical diagnosis, etc.) require addressing multiple user goals that may often conflict with one another [40]. Often these goals are specified to the system using a multi-objective objective function representation [41], [42]. In such objective functions, it is considered as 'k' (number of objectives) increases, the power of finding dominant solutions diminishes because satisfying each of the objectives becomes mathematically intractable [43]. Purshouse et al. confirmed that multiple objectives may be conflicted with one another; resolving conflict may show better performance in one objective than others [44]. Zhang et al. defined conflict analysis as a method to find conflicts, reason about it, and then resolve it [9]. Bell et al. further described conflicts in decision making and summarized quantitative approaches to address them in optimization problems [10]. Reed et al. explored scatterplot charts to visually inspect the set of conflicting objectives to solve a ground water optimization problem [11] (e.g., discovered a conflict between cost and uncertainty). There is a recent interest in multi-objective machine learning optimization functions, which tackles conflicting user objectives. For example, in model selection, there is the conflict between model complexity and model accuracy (more complex, more accurate the model) [8], [45]. Multi-task learning is another avenue where multiple tasks are solved jointly using a multi-objective optimization paradigm; however these tasks often conflict with one another that needs a trade-off analysis [7]. A common solution is to utilize a proxy objective to minimise a weighted linear combination (per task) of loss. Sener et al. showed a solution to the conflicting objectives by solving for pareto optimal solutions [7]. While previous work in conflict resolution used elementary visualisations, none of these ventures allowed users

to detect and resolve conflicts in objective functions for classifier construction using Auto-ML, a problem that we solve in this paper.

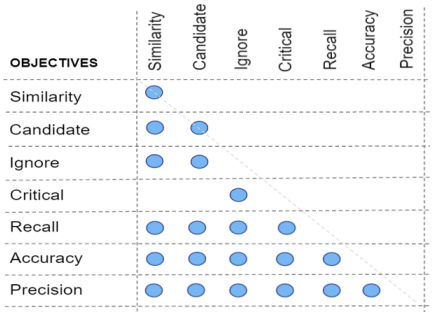

Fig. 2. Conflict matrix, shows conflict possibilities between objectives.

The objective function is a means to maximize (or minimize) something. This something is a numeric value. In the real world it could be the cost of a project, a production quantity, profit value, or even materials saved from a streamlined process

3 Types of Conflicts in Objective Functions

Objective functions are used to minimize or maximise a user-defined goal expressed numerically. In this work, we consider objective functions that allow selection of optimal classifiers from an Auto-ML system (e.g., Hyperopt [46], Optuna [47]). In this, the goal is to maximize an objective score, which is a weighted linear combination of many objectives. An example of a user-defined objective can be that the model: (1) correctly predicts a specified number of critical data instances or (2) correctly predicts a certain class label or (3) predicts user-defined similar data instances in the same class group. When a model is trained the objective function is utilized to compute the score based on the predictions made by the model. Here the objective functions described are not the objective functions that drive the decision making processes in ML algorithms such as SVM, Logistic Regression, or Neural Networks. This renders our approach model-agnostic, empowering end users to test plethora of objectives as they find optimal models.

In creating these objective functions, users may inadvertently define conflicting objectives. For example, users may specify a subset data instances that should be predicted in the same class, while also specifying a subset of these instances as good representative examples of two different classes. In order to understand what conflicts between user-specified objectives are, we adopt the objective categories as defined by Das et al. [6]. There were four categories: (1) *Instance-based*, (2) *Feature-based*, (3) *Train-objectives*, and (4) *Test-objectives*. Within each of these categories there are a set of objectives such as *Candidate*, *Similarity*, *Ignore*, etc. under the *Instance-based* category. Explaining each of these categories is beyond the scope of this paper. We categorized these conflicts as:

1) **Conflicts based on choices:** Conflicts can be categorized based on users' subjective choices, presuming these choices follow best ML practices (e.g., guarding against overfitting, constructing classifiers that represent every class precisely).
   Logic based conflicts: These are conflicts that are logically incorrect but does not violate best ML practices, which we term as *logic-based-conflicts*. For example, a user may specify a set of data items to be part of the objective *Similarity* (where these data items are expected to be in the same class label say A), while specifying a subset of these data items as *Candidate* (an objective specifying data items that are good representative examples of a class label) of a different class say B. These conflicts are logically incorrect but do not violate best ML practices. Refer Figure 2 to see every conflict combinations.
   ML practices-based conflicts: There can be conflicts which are not logically incorrect but may be violating best ML practices. For example, building a classifier using only *Train-objectives*, and not *Test-objectives* as a constraint, may produce overfitted models that are not generalizable to unseen data. Similarly for an imbalanced data if only an *Accuracy* objective is specified as opposed to *F1-Score* or *Precision*. This may trigger the model solver to select classifiers that performs poorly on minority class labels. In this paper we are not addressing conflicts that may occur due to violating best ML practices. We realized that handling these type of conflicts requires lot of work, which might be a research project in itself.
2) **Conflicts based on time of occurrence:** *Logic-based-conflicts* can be further categorized based on when they occur in the modeling pipeline.
   Before model conflicts: Some conflicts can be computed before even training a model, while a few other conflicts can be only ascertained after a model is constructed. The first kind, which we term as *Before-model-conflicts*, can be automatically computed before a ML model is constructed based on the objectives in the interactive objective function. For example, a user has specified a set of data items - I that should be placed in the same class label, say *Dog*, while also specified a subset of I, say J data items as *Candidate* objective for the class label *Cat*. Here *Candidate* objective means data items that are strong representative of a specified class label. So the conflict is that the same data items are specified as examples of two different label categories *Dog* and *Cat*. Similarly another example of a *Before-model-conflict* is between the objectives *Critical* and *Ignore*. *Critical* represents a set of data items that are very important for the user, and thus the user expects the model to predict them correctly, and *Ignore* represents a set of data items that the user deems unimportant, or noise or garbage from the training set. A user may specify a set of data items as *Critical*, while in a future iteration of the model construction may specify a subset of *Critical* data items as *Ignore*. Such kinds of conflicts can be computed before a model is constructed.
   After model conflicts: In addition to the above, there may be other types of conflicts that are only noticed when a model is constructed or many iterations of model construction have occurred which we term as *After-model-conflicts*. From the literature we know that in a multi-objective objective function, all objectives cannot be attained [8]. In such scenarios, a set of pareto-optimal solutions are presented to the user in which only a subset of objectives are attained in each of the pareto-optimal solutions [37]. Analysing model log data over multiple modeling iterations, the system can infer which objectives are repeatedly unattained, or which set of objectives cannot be solved together (at the same time). In such cases, these objectives are in conflict with one another, meaning that the specification of one, blocks attainment of the other or vice versa in the objective function [9]. For example, the system may infer that a highly weighted *Train-Accuracy* objective is prohibiting the model solver to find a model that also attains the *Similarity* objective successfully. These conflicts can only be inferred when the model is constructed or when multiple iterations of model construction has occurred.

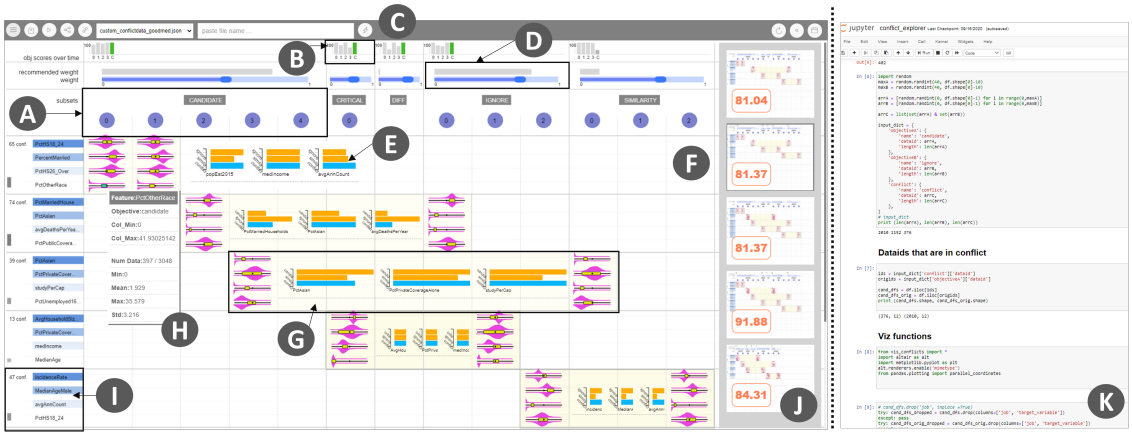

Fig. 3. The CACTUS system - A. Data subsets per objective. B. Model spark bars. C. Control bar. D. Gray bars show recommended objective weights. Blue sliders allow users to control weights per objective. E. Variance bars. F. Conflict view. G. Conflict box. H. Tooltip from a whisker box. I. Top 4 highly variant attributes. J. Objective function gallery (can be placed to the left or right). K. Jupyter notebook.

We scope our conflict detection and resolution method to handle conflicts that are logically incorrect (from the Conflicts based on choices) and of the type *Before-model-conflict* (from the Conflicts based on time of occurrence), but do not violate best ML practices. As the *after-model-conflict* investigation needed more intensive engineering effort, we plan to address it in the future.

4 Tasks and Design Guidelines

Grounded on previous work and our experience with interactive objective functions, we formulated the following analytical tasks:
AT1: Import and visualize objective functions from a Jupyter notebook (NB). Also, allow users to switch back and forth between multiple objective functions that they import in their analysis.
AT2: Inspect conflicts between objectives to address problems with an objective function. Compare severity of conflicts between any objectives in a function, and between multiple objective functions.
AT3: Resolve conflicts by re-assigning conflicted data items to a chosen objective. Control implication of any conflict by adjusting the respective objectives' weights. Export conflicted data items' ID or the entire objective function to NB to redefine objectives.
AT4: Incrementally construct many ML models and inspect model trade-offs between multiple model alternatives, through the interactive creation of objective functions. These functions can be variants of the function that they import, created in the process of conflict resolution, re-assigning weights or re-writing the function in NB.
We set forth the following design guidelines for CACTUS:
DG1: CACTUS should visualize an objective function and assigned weights to its objectives. Its interaction should allow users to create variants of the function. [**AT1**]
DG2: CACTUS should show performance of a selected model on each of the specified objectives for every iteration of model construction. Users should be able to visually perceive how the model satisfied the specified objectives. [**AT4**]
DG3: CACTUS should visually show conflicts between objectives. Users should be able to visually understand the conflicts and seek details on each conflict (e.g., distribution of the conflicted data, see variance of attributes). [**AT2**]
DG4: CACTUS should help users to interactively resolve conflicts. Furthermore, it should provide affordances to either perform conflict resolution in the interface or allow users to export conflicts to NB (to redefine objectives). [**AT3**]

5 CACTUS: System Design

CACTUS's frontend was deployed with webGL based PhaserJS gaming library. The backend was a NodeJS server running python code. To interface with Jupyter notebook, we utilised the nbconvert package [48] in python. Auto-ML model solvers were simulated using Optuna [47] package in python. The views of CACTUS are: (1) Conflict view, (2) Model spark bars, (3) Venn diagram, (4) Feature plots, (5) Objective function gallery.

5.1 User Interface

Conflict view: This view shows conflicts using a table representation, where every column shows an objective from the loaded objective function (see Figure 3-F, **DG1**). The second row (with the blue numbered circles, see Figure 3-A) represent subset data instances that were specified as examples as part of the respective objective. Every row in the table encodes a conflict (**DG3**) between a pair of objectives (e.g, *Candidate*, and *Ignore* in second row). Conflict pairs are emphasized by a highlighted rectangular box (also called *Conflict box*, see Figure 3-G and Figure 4-H), where 4 most highly variant attributes are vertically ordered (see the blue gradient text on the left in Figure 3-I). Aligned with each of these attributes, a violin plot, with a whisker box plot is rendered (Figure 3-G and Figure 5). Here the violin plots show the distribution of the data in relation to that attribute, and the whisker plot shows the shape of the data instances that are part of the objective. Further, users can hover their mouse on the whisker box to see detailed information about the shape of the data (Figure 3-H, **DG3**). Thus, using this visual technique, users can compare the conflicted data instance's shape across two objectives that are in conflict with one another. The Conflict box, also shows a horizontal bar chart of *Variance bars*, where it shows the variance of the data in the two objectives (shown in orange) and the variance of the conflicted data items (shown in blue) in relation to top 3 highly variant attributes (see Figure 4-D and Figure 3-E). Using this view, users can understand how similar or different are the conflicted data in comparison to the data items that are part of the two objectives.
Model spark bars and weightings: On top of the Conflict view, CACTUS also renders horizontal sliders (shown as blue bars in Figure 3-D) that allow users to interactively specify weights to the objectives (**DG1**, **DG2**). In addition, users can refer to system recommended weights, if they are unsure about the weights for each objective (shown as gray bars in Figure 3-D). CACTUS allows users

to incrementally train models while they adjust the objective function (e.g., by resolving conflicts or updating objective weights). Per iteration when a new model is trained, its validation accuracy is plotted as a series of vertical bars, called *Model spark bars*, scaled between 0 to 100 as seen in Figure 3-B. These bars allow users to compare performance of the model with respect to specific objectives.

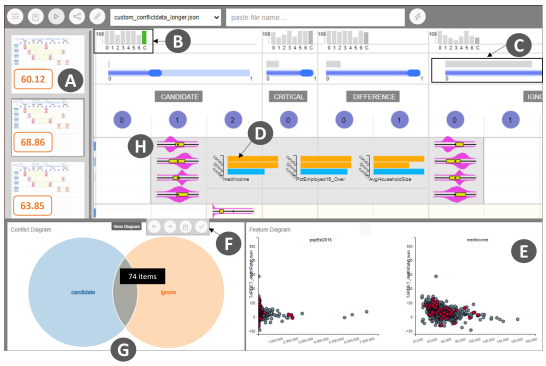

Fig. 4. A. Objective function gallery. B. Model spark bars. C. Gray bars show recommended weights. Blue bars are sliders to allow users to specify weights. D. Variance bars. E. Feature plots.F. Context menu. G. Venn diagram view. H. Conflict box.

Venn diagram view: As users explore the conflicts, they can click on the *Conflict box* between an objective pair, to trigger the system to open the bottom tray. It reveals a venn diagram showing the overlap between the pair of objectives. The size of the circles reflect how many examples comprise each objective, while the overlap of the circle encodes the conflicted data items (see Figure 4-G) that are shared by both objectives. Furthermore, users can hover over the overlapped region to see the distribution of the data on the *Feature plots* (Figure 4-E) and also on the *Variance bars* (see Figure 4-D and Figure 5). Based on their exploration, they may decide to resolve the conflicts (**DG4**) by either: (1) Moving the points to the left or to the right objective, (2) Exporting the conflicted data to the Jupyter notebook, and (3) Completely removing the conflicts from the objective function using the context menu seen in Figure 4-F. Resolving any conflict, removes the visual representation of it, i.e., a single row from the *Conflict view* is removed from the display.

Feature plots: This view plots a set of k attributes with highest variance as small multiples of scatterplots (Figure 4-E). Each of these charts are plotted with the target variable (or dependent variable) on the y axis. The design of this view is intended to show users the shape of the data in an objective or in a conflict in relation to the input data (**DG4**). This view is linked with the *Venn diagram view*.

Objective function gallery: As users resolve conflicts or change weights of objectives, a new version of the objective function is stored in the memory. Users can access the history of objective function creation through this view, where each state of the objective function is shown using a thumbnail preview (see Figure 3-J and Figure 4-A, can be placed to the left or the right of the Conflict view). The preview also shows the trained models' validation accuracy score scaled between 0 to 100 (**DG2**). This view also supports users to go back to a previous state of the function.

5.2 Conflict Detection and Resolution Method

Next we explain how CACTUS detects conflicts:

Conflict parser: A user may write an objective function, O in Python/Jupyter notebook comprising of a set of k objectives $\omega_1, \omega_2, \omega_3, ..., \omega_k$. Furthermore, each of them can be specified with a set of k scores $(s_1, s_2, s_3, ..., s_k)$. Thus, O can be represented as a weighted linear combination of these objectives as seen here, $O = s_1 * \omega_1 + s_2 * \omega_2 + s_3 * \omega_3 + s_4 * \omega_4 + s_5 * \omega_5$. The objectives checks, if a set of user-defined data instances are correctly predicted by a classifier. Thus objective ω_i is represented as a set of training data instance T ID's as $t_1, t_2, t_3 ....t_l$. However, in the case of *Candidate* objective, they may also be stored as ID's for a specific class label L_1, L_2,...L_b, (b class labels). The conflict parser module of CACTUS checks for any overlap between all the paired combination of objectives from O. For example, it utilizes T_i and T_j between the objectives ω_i, ω_j, using the function $FN(\omega_i, \omega_j)$ to find conflicted data ID's T_c. Finally, this module generates a hashmap object F, where keys are hashed to represent the objective pairs $(\omega_i - to - \omega_j)$, and the values are the conflicted data ID's T_c.

Data distribution and variance: Conflicts are visualized using the F object (generated by the *conflict parser* module). Sequentially, the system tracks the pair of objectives P_i that are in conflict using the hash-keys (f_k) of the object F. Next, it retrieves the set of data ID's T_i that are specified as examples as part of the objective pairs P_i. It also recovers T_j from F that represents the conflicted data items. Using T_i, the system first retrieves the top 3 (this number can be adjusted) attributes with highest variance in this set. Next it draws the violin plot V and the whisker plot W. While V shows the distribution of the full training set, the whisker plot W, shows the distribution of the examples that are part of the respective objectives in P_i. Similarly, the *Variance bars*, are rendered to visualise the variance for the data ID's in P_i, and the conflicted data items T_j. The venn diagrams are also drawn using the object F.

Model solver: Users can also utilize any Auto-ML model solver M to sample n classifiers C (e.g., can be adjusted $n = 200$) in Python or in a Jupyter notebook. For the current prototype we tested with Hyperopt [46], and Optuna [47] but can be replaced by Auto-SKLearn [49], or Auto-Pytorch [50] if required. In this pipeline, M expects an objective function O, to score each of the classifiers c_i in C. The highest scoring (H) classifier c_k is selected and its' overall performance (e.g., validation accuracy score), and per objective performance is visualized in CACTUS. While the model construction part is handled by Python or the Jupyter notebook, CACTUS only ingests the objective function O, and visualizes conflicts.

Weight recommendation: The set of weights $S = s_1, s_2, s_1 ....s_k$ in O can be specified from the Jupyter notebook. These can also be interactively adjusted from the interface of CACTUS using the sliders (Figure 4-C). To further guide users, our approach also recommends weights S' (between 0 and 1, for each objective ω_i). In the initial iterations, the recommendations are randomly initialized, however, as users incrementally construct multiple versions of objective functions $(O = O_1, O_2, O_3, ...O_f)$ and models $(M = M_1, M_2, M_3, ...M_f)$, these weights are recommended based on the probabilistic likelihood of the weight settings that found success in: (1) Maximising the overall model accuracy, and (2) Maximising the objective score for which the weight is recommended. We followed the Bayesian approach [49] in modeling the probabilistic likelihood of the weight settings.

6 Usage Scenario

Here we explain a scenario where an analyst constructs a classifier that performs optimally on domain-specific objectives. Consider James is an analyst in the public policy department of US Government seeking to construct a classifier to predict *Cancer mortality rate*. He has access to a *Cancer mortality dataset (per US*

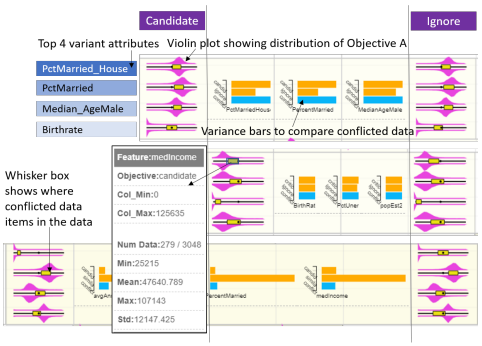

Fig. 5. Violin plots and whisker boxes show how conflicted data items are similar or different to the pair of objectives.

County) [51] containing 3048 records/rows; each row represents a US county. It contains 34 independent variables such as *incident-rate*, *median-age*, *avg-household-size*, *birth-rate*, and others. James wants to predict the *Death-Rate-Per-County*, annotated by the labels: *negligible*, *low*, *medium*, *high*, and *very-high*. James uses Jupyter notebook (NB) to explore and analyse the data (Figure 3-K). To classify the data, James trains a gradient boosted classifier in NB and observes a relatively poor accuracy of only 72%, and 65% on the training and validation set respectively. Motivated to select a preferred optimal model for this problem, he decides to use Optuna - an Auto-ML solver [47], and writes a custom objective function for this package to find optimal classifiers in NB. James re-trains a new classifier by feeding this objective function (with objectives *Candidate*, *Ignore*, and *Critical*) in Optuna, and notices that the training accuracy improved to 86%, while the validation set improved only marginally (69% accuracy). In their objective function specification, James filters the data with less than value 50000 of the attribute *med-income* and higher than 18% of the attribute *poverty-percent* to be classified in the same class label of *medium*, as he thinks these data samples should be similar. Furthermore, he selects a set of counties whose *birth-rate* and *median-age* attribute values range between 6 to 15% and 35 to 50 respectively, as critical counties that should be correctly classified.

Motivated to explore different variations of this objective function and to resolve any conflicts between the specified objectives, James imports it in CACTUS (Figure 3). James sees the conflicts in the *Conflict view*, and looks at the left end of each row to find the most severe conflict (Figure 3-I). He notices that there are 74 data samples that are in conflict between the objectives *Ignore* and *Candidate*. He also inspects the conflicted data items' position with respect to the attribute values shown in the violin plots and whisker boxes (Figure 5). James notices that the examples shown as part of the objectives have similar shape and thus likely confusing the model solver. He clicks on its *Conflict box* to reveal the *Venn diagram view* (see Figure 4-G) showing the conflicted data samples. Hovering over each of the venn diagram's arc sectors, James inspects the data samples on the *Feature plots* that shows the relationship of top 2 highly variant attributes, plotted on the x-axis with the target variable on the y-axis. This view answers if these samples are similar or different from the two objectives (as seen in the red dots in Figure 4-E). James realizes that a condition to specify a set of data samples with attribute values of *med-income* lesser than 50000, might have caused this conflict. He exports this conflict to NB to further re-define the objectives *Ignore*, and *Candidate*. After correction, James adds two more objective subset examples for: (1) *Similarity*, and (2) *Critical* in NB. James trains a new model, but finds that many important counties are incorrectly classified by this model. He re-defines the *Similarity* and the *Candidate* objective and retrains a set of new models in NB.

Upon importing the new function to CACTUS, James observes that one of the most severe conflict is between *Critical* and *Similarity*. To remove this conflict, he filters the conflicted data samples using the data ID's retrieved from CACTUS in NB. He finds similar data samples (using cosine distance) to the conflicted data samples from the training set for the *Critical* objective. James replaces the similar data samples in the *Critical* objective to resolve this conflict. Next, he trains a new classifier, loads it in CACTUS to see that the *Validation-set* accuracy improved to 94.58%. Happy with the analysis results, James exports the best model and the final objective function to share with their collaborators.

7 Evaluation

We conducted a within-subject user study of CACTUS to validate the effectiveness of our technique. Considering the lack of any other visual interface similar to CACTUS, we designed our study to address the following research questions. As an alternative, the command line interface was not considered as we were investigating a joint 'code' and 'no-code' approach as used in CACTUS.

RQ1: Does CACTUS make it easy to precisely find conflicts between objectives in objective functions for classifiers?

RQ2: Does CACTUS help users in correctly resolving conflicts?

RQ3: Does CACTUS support users to compare objective functions and understand trade-offs between them?

We recruited 14 participants (9 Male, 5 Female), aged between $22-36$ ($M=26.06$ $[22.41, 29.71]$), by inviting participants through our university mailing lists. Our requirement was that they should know how to read/write basic python code, with elementary understanding of classifier construction and exploratory data analysis. Our participants were a mix of Masters and PhD students from computer science, analytics, geography, and urban planning. They had basic familiarity with data analysis ($M=5.26$ $[3.73, 6.79]$, on a Likert scale rating of $1-7$, higher is better), and basic ML expertise ($M=4.85$ $[3.63, 6.07]$). The study was conducted remotely using Bluejeans [52] in light of the on-going COVID-19 pandemic. It lasted 60-70 minutes and at the end of a successful session we compensated participants with a $10 Amazon gift card. The system was deployed on our computer, which we shared with participants using a publicly accessible URL retrieved using NGROK [53]

7.1 Study Design and Datasets

We began the study with a live demo, showing participants how CACTUS works and how its visualizations can be interacted with. We also demonstrated how the system integrated with a Jupyter notebook environment to seek objective functions (pre-defined by writing Python scripts), and data instances. During this session, we encouraged participants to ask as many questions they wanted to clarify any confusion with respect to the workflow or the system interface. Next, when we felt confident that participants were ready for the tasks, we proceeded to the experimental sessions. To answer the previously mentioned **RQ**'s we considered these dependent variables: (1) *Task completion times* to detect/find and resolve conflicts, (2) *Conflict resolution success rate*, i.e, the number of conflicts the participants correctly resolved out of the total conflicts for all the given objective functions (between 0 to 1), (3) *Model Accuracies*, accuracy score of models per iteration of objective function specification (scaled between 0 to 1), (4) Number of iterations as users

incrementally created objective functions by resolving conflicts, and (5) *User preference ratings* that includes *Ease of use*, *Intuitiveness of the GUI*, *Steep learning curve*, and other relevant system interactions (all of the scores were normalized between 0 to 1).

For the practice session, we provided a dataset of 5000 IMDB movie records [54]. The data had attributes such as *gross-revenue*, *budget*, *cast-facebook-likes*, *number-user-votes*, etc. It was a multi-class classification task to predict the rating of a movie between *low*, *moderate*, *high*, and *very-high*. For the first experimental session, we provided San Francisco city's employment dataset [55] containing 25000 records of job types for the quantitative evaluation. Each row in the data contained information about a job's remuneration information containing attributes such as *dental-benefits*, *annual-salary*, *health-benefits*, *retirement-compensation*, etc. The task was to predict the job's department which had 5 classes e.g., *Cultural/Recreation*, *Public Service*, *Healthcare*, *Administration*, and *Other*. For the next experimental session, we provided the *Cancer mortality dataset (per US County)* [51] to predict *Death-Rate-Per-County*. The class labels were Very high, *High*, *Moderate*, *Low*, and *Negligible*. The dataset contained 3048 rows, each row representing the death rate of a US county and was annotated with one of the five categories of class labels. Furthermore, it had 34 attributes (1 categorical variable) including *incident-rate*, *median-age*, *avg-household-size*, *birth-rate*, and others.

7.2 Tasks and Procedure

In the practice session we provided users with a list of 3 pre-defined objective functions (written in a Jupyter notebook) on the IMDB movies dataset [54]. We asked them to load them in CACTUS and visually explore the conflicts in various objectives. Next after 15 minutes of practice we asked them questions such as: (1) Which objective pair has the highest conflict? , (2) Name the top 2 highly variant attributes for the conflict between *Ignore* and *Similarity*, (3) Resolve conflicts between *Similarity* and *Candidate*. For session *A* (low-level analytic tasks 1-6) we asked participants to load three objective functions (pre-defined by us in a Jupyter notebook) on the San Francisco's salary dataset [55] and perform these tasks (randomized to remove learning effects):

Task 1: Report number of data items that are in conflict between *Ignore* and *Candidate* from the second objective function.

Task 2: Name the top 2 attributes with high variance between the objectives *Similarity* and *Candidate*.

Task 3: Resolve the first and the last conflict from the third objective function. Train (and export) a new model after you resolve each conflict and then compare the model performance.

Task 4: Out of the three objective functions, find the function that has the highest conflict between any of the objectives.

Task 5: Train three models by changing weights of any of the objectives for each of the three objective functions. Which objective function found the better performing model?

Task 6: Export any conflict from the top 2 best objective function from the set of saved functions to Jupyter notebook.

Task 7: In session B (high-level exploratory task), we asked participants to freely use CACTUS and improve model performance in 8 mins. using a pre-defined objective function and a baseline classifier on the Cancer mortality dataset [51].

7.3 Data Collection

We captured video and audio of participants screen. We saved log data containing selected models' learning algorithms, and hyperparameters, predicted class labels, interacted objectives and conflicts. When participants completed all the tasks for both sessions, we asked them to fill a NASA-TLX form [56], and a post-study questionnaire with a set of Likert scale questions (in a 7 point scale). In the end we conducted a semi-structured interview asking open-ended questions about the workflow, system usability, and interaction design of the interface. Throughout the tasks we encouraged participants to think aloud as they interacted with CACTUS.

7.4 Quantitative Analysis

We broadly measured if using CACTUS: (1) users can detect conflict easily and successfully, (2) users can resolve conflicts with precision, and (3) users can compare objective functions over time and learn trade-offs between them. Thus, we validated CACTUS's success based on the following quantitative metrics:

Task completion times: We measured task completion time when users were asked to: (1) Report a conflict between a pair of objectives ($M = 2.43 mins.$ $[1.41, 3.45]$), (2) Report highest conflict between three pre-defined objective functions ($M = 5.12 mins.$ $[2.81, 7.43]$). Next, we measured task completion time when participants: (1) Resolved conflicts between a pair of given objectives in an objective function ($M = 3.03 mins.$ $[2.59, 3.47]$), and (2) Resolved conflicts across the three objective functions ($M = 5.02 mins.$ $[4.00, 6.04]$).

Correctness in conflict resolution: In addition, 13 out of 14 participants successfully found the right conflicts between objectives. We observed 92.86% success rate among participants to find conflicts in an objective function. We also observed that every participant but two was able to successfully resolve conflicts (85.17% success rate). The two failed to complete the task due to a bug (latency in recording conflict changes by the user) in the system, that we plan to fix in future.

Incremental comparison of objective functions: We investigated log data to assess if participants were able to compare objective functions and learn trade-offs between objectives as they trained multiple classifiers. In doing so, we observed that on an average, participants iterated 10.23 times ($M = 10.23$ $[6.69, 13.77]$) to improve the given classifiers' baseline validation accuracy score of 78.24% on the Cancer mortality dataset [51]. 11 out of the 14 participants selected an objective function (to export, as their final selection) from a previous iteration in time. Furthermore, we observed a set of approaches to train new models: (1) Train a new classifier by changing weights only ($2/14$ participants), (2) Train a new classifier by resolving conflicts only ($4/14$ participants), and (3) a hybrid approach of the two which was the most popular ($8/14$ participants). We found that 88.34% of the participants were successful in improving the baseline classifiers' performance using these approaches.

User preference ratings: We analysed Likert scale user preference ratings (on a scale of 1 to 7, higher is better) after users interacted with CACTUS. Participants expressed that CACTUS was *Easy to use* ($M = 6.04$ $[5.85, 6.23]$), and its interface was *Intuitive* ($M = 5.51$ $[5.05, 5.96]$). The majority also confirmed that the interface did not have a *steep learning curve* for new users ($M = 2.45$ $[1.25, 3.65]$), lower is better in this case). Most of the participants felt that they were successful in resolving conflicts ($M = 5.11$ $[4.95, 5.27]$), and they were able to incrementally improve models' validation accuracy ($M = 5.23$ $[5.01, 5.45]$). Furthermore, the participants confirmed that the *Conflict view* was expressive to not only help them find conflicts but also know which conflicts were more severe from the set ($M = 6.03$ $[5.54, 6.52]$).

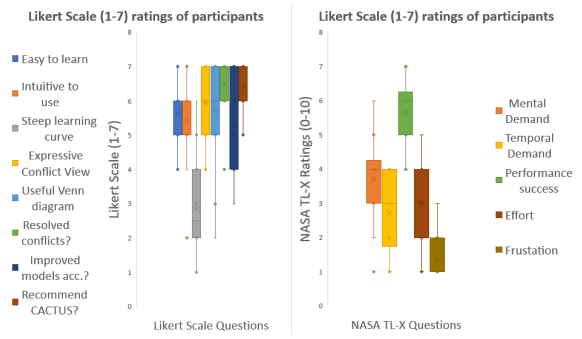

Fig. 6. Results of Likert scale ratings and NASA TL-X scores for CACTUS.

Likewise the *Venn diagram view* was found to be very useful to resolve conflicts ($M = 5.32$ $[5.11, 5.53]$). However, many participants found the *Feature plots* could have been more useful, if they knew the data better ($M = 3.11$ $[1.55, 4.67]$). Next, from the NASA-TLX survey we observed that on average every participants' mental workload, and frustration towards the tasks were on the lower side ($M = 2.11$ $[1.01, 3.21]$ out of a 10 point scale; lower is better).

7.5 Qualitative Analysis

Expressive visualisations: Most of the participants liked the visual representations used in CACTUS to represent conflict. In addition, they liked the current interaction design to help them resolve conflicts. **P10** noted, *"Easy to use, UI was straight forward. Resolving conflicts was easy. I liked that I could do things without much interruption."*. However, some participants expected to adjust Python code from CACTUS without transitioning to the Jupyter notebook.
Incremental objective function exploration: We observed that participants enjoyed the workflow of incrementally changing objective functions to train better performing models. They either changed weights or they resolved most severe conflicts to create a variety of objective functions. **P04** expressed *"I liked the plot on the top that shows you the history of the model performance, it was very helpful for me to keep track of the model improvement."*. A few participants gamified the model training process, e.g., **P03** said *"I liked the highlighted bar charts showing the attributes with the highest variance. That helped me tweak my weights as well as resolve conflicts, to beat the machine to find a better [performing] model"*.
Glean insights about the process: A few participants expressed the desire to learn a bit more about the process, in addition to be able to interactively train models. For example, **P02** shared *"At times it was hard to know how to change the weights and whether to move left or right when trying to resolve conflicts. I expect to see more visual cues on how to improve models' accuracy."*. In future, we plan to make this incremental modeling process more transparent using GUI elements within the interface.
Conflict resolution strategy: Broadly, we understood two main strategies that the participants used to resolve conflicts. The first approach was to compare the shape of the data in conflict, to the data in the two objectives by observing the *Feature plots*. Based on the shape resemblance they moved the conflicted data instances to one of the objectives. The other approach was to look at the **Variance bars** to find resemblance of the conflicted data to one of the objectives. When participants were not sure, how to resolve the conflict, they preferred to export it to Jupyter notebook to provide better examples by writing Python code. **P08** said, *"My first step would be to find which variables are useful. I would then, look at the bar charts to see if the conflicts were closer to one of the objectives. If not, I would look at the scatterplot and make judgements by approximating the average location of each of the three groups."*

8 Discussion and Limitations

Collaborative objective functions: In ML teams, objective functions are often designed by multiple members of the team to solve the same problem. In situations like this CACTUS can helps visually compare multiple versions of objective functions from different team mates, and then explore the space of possible objective functions and conflicts that may arise from each. That may open a new research area as an extension to our current work, in which users may like to resolve conflicts by collaboration across different geo-locations, or from different devices.
Gamification of modeling: We observed a few participants gamified the process of model construction, by testing various weight settings of objectives, resolving conflicts, or specifying new objectives. Their goal was to beat the previous models' performance score and also to perform better than the model constructed using system recommended weights. Many participants elicited there is a lot of value in this, as not only it helps them ideate and explore faster, but also makes the process of finding a suitable model more fun. In future, we see lot of potential in supporting the interface with features that further encourage and guide users in this process of gamified model creation.
Limitations in conflict resolution: From the study, we realized there are several aspects of CACTUS that can be improved. For example, we found the current interface showing the incremental view of objective functions (*Model spark bars* and the *Objective function gallery*) is often difficult to track because it only supports sequential record. Every change in one parameter leads to a model retraining, and saves a new copy of the function. This may be difficult to track when users have iterated many times. We plan to research further to find potentially better design choices in-terms of visual design and interaction. Another issue, we realise is that data exploration is critical to resolve conflicts and design good objective functions. A few participants in the study, who were less versed with the data, felt uninformed to change the weights and whether to move left or right when resolving conflicts. In CACTUS, we focused on conflict resolution and objective function comparison, while separated the task of data exploration in Jupyter notebook.
Scalable conflicts: The current prototype's implementation is designed to be model agnostic, meaning it can work with classification, or regression models (supervised ML). It can also work with sequence data such as text, time-series and image data. Currently, the interactions are tested on medium sized data sets (i.e., with data samples in the range of 100k). Though we tested with multiple conflicts per row, we discarded the approach, as it may lead to ineffective selection of conflict intersections in the venn diagram view.

9 Conclusion

We presented a novel VA system CACTUS, that interactively helps in detecting and resolving conflicts among objectives in a multi-objective objective function. In this paper, we also discussed a list of various types of conflicts that may occur when users interactively specify objective functions to Auto-ML model solvers to classify tabular data. With a quantitative and qualitative user study we show that our technique helps users to interactively detect and resolve conflicts between objectives, and incrementally train ML classifiers in tandem with a Jupyter notebook environment.

REFERENCES

[1] S. Amershi, M. Chickering, S. Drucker, B. Lee, P. Simard, and J. Suh, "Modeltracker: Redesigning performance analysis tools for machine learning," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2015)*, April 2015.

[2] B. C. Kwon, B. Eysenbach, J. Verma, K. Ng, C. D. Filippi, W. F. Stewart, and A. Perer, "Clustervision: Visual supervision of unsupervised clustering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 142–151, Jan 2018.

[3] S. Das, D. Cashman, R. Chang, and A. Endert, "Beames: Interactive multimodel steering, selection, and inspection for regression tasks," *IEEE Computer Graphics and Applications*, vol. 39, no. 5, pp. 20–32, Sep. 2019.

[4] S. L'Yi, B. Ko, D. Shin, Y.-J. Cho, J. Lee, B. Kim, and J. Seo, "Xclusim: a visual analytics tool for interactively comparing multiple clustering results of bioinformatics data," *BMC Bioinformatics*, vol. 16, no. 11, p. S5, Aug 2015. [Online]. Available: https://doi.org/10.1186/1471-2105-16-S11-S5

[5] H. Piringer, W. Berger, and J. Krasser, "Hypermoval: Interactive visual validation of regression models for real-time simulation," in *Proceedings of the 12th Eurographics / IEEE - VGTC Conference on Visualization*, ser. EuroVis'10. Chichester, UK: The Eurographs Association & John Wiley & Sons, Ltd., 2010, pp. 983–992. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2009.01684.x

[6] S. Das, S. Xu, M. Gleicher, R. Chang, and A. Endert, "Questo: Interactive construction of objective functions for classification tasks," *Computer Graphics Forum*, vol. 39, no. 3, pp. 153–165, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13970

[7] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 527–538. [Online]. Available: http://papers.nips.cc/paper/7334-multi-task-learning-as-multi-objective-optimization.pdf

[8] Y. Jin, *Multi-Objective Machine Learning*. Springer, 2006.

[9] L. Zhang, C. F. Madigan, M. H. Moskewicz, and S. Malik, "Efficient conflict driven learning in a boolean satisfiability solver," in *Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '01. IEEE Press, 2001, p. 279–285.

[10] . R. H. Bell DE, Keeney RL, *Conflicting Objectives in Decisions*. John Wiley and Sons, 1977.

[11] P. M. Reed and B. S. Minsker, "Striking the balance: Long-term groundwater monitoring design for conflicting objectives," *Journal of Water Resources Planning and Management*, vol. 130, no. 2, pp. 140–149, 2004.

[12] H. Li, S. Fang, S. Mukhopadhyay, A. J. Saykin, and L. Shen, "Interactive machine learning by visualization: A small data solution," in *2018 IEEE International Conference on Big Data (Big Data)*, Dec 2018, pp. 3513–3521.

[13] B. Alsallakh, A. Jourabloo, M. Ye, X. Liu, and L. Ren, "Do convolutional neural networks learn class hierarchy?" *CoRR*, vol. abs/1710.06501, 2017. [Online]. Available: http://arxiv.org/abs/1710.06501

[14] M. Liu, S. Liu, H. Su, K. Cao, and J. Zhu, "Analyzing the noise robustness of deep neural networks," *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 60–71, 2018.

[15] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. F. Lelieveldt, E. Eisemann, and A. Vilanova, "Deepeyes: Progressive visual analytics for designing deep neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 98–108, Jan 2018.

[16] O. Biran and K. McKeown, "Human-centric justification of machine learning predictions," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, pp. 1461–1467. [Online]. Available: http://dl.acm.org/citation.cfm?id=3172077.3172090

[17] K. Patel, N. Bancroft, S. M. Drucker, J. Fogarty, A. J. Ko, and J. Landay, "Gestalt: Integrated support for implementation and analysis in machine learning," in *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '10. New York, NY, USA: ACM, 2010, pp. 37–46. [Online]. Available: http://doi.acm.org/10.1145/1866029.1866038

[18] S. Das and A. Endert, "Legion: Visually compare modeling techniques for regression," in *2020 Visualization in Data Science (VDS)*, 2020, pp. 12–21.

[19] Y. Sun, E. Lank, and M. Terry, "Label-and-learn: Visualizing the likelihood of machine learning classifier's success during data labeling," in *Proceedings of the 22Nd International Conference on Intelligent User Interfaces*, ser. IUI '17. New York, NY, USA: ACM, 2017, pp. 523–534. [Online]. Available: http://doi.acm.org/10.1145/3025171.3025208

[20] T. Li and T. Zajonc, "Hypertuner : Visual analytics for hyperparameter tuning by professionals," 2018.

[21] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau, "Activis: Visual exploration of industry-scale deep neural network models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, pp. 88–97, 2017.

[22] A. Chatzimparmpas, R. M. Martins, I. Jusufi, K. Kucher, F. Rossi, and A. Kerren, "The state of the art in enhancing trust in machine learning models with the use of visualizations," *Computer Graphics Forum*, vol. 39, no. 3, pp. 713–756, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14034

[23] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu, "A survey of visual analytics techniques for machine learning," *Comput. Vis. Media*, vol. 7, no. 1, pp. 3–36, 2021. [Online]. Available: https://doi.org/10.1007/s41095-020-0191-7

[24] K. Patel, S. M. Drucker, J. Fogarty, A. Kapoor, and D. S. Tan, "Using multiple models to understand data," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, ser. IJCAI'11. AAAI Press, 2011, pp. 1723–1728. [Online]. Available: http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-289

[25] X. Z., A. Singla, S. Zilles, and A. N. Rafferty, "An overview of machine teaching," *CoRR*, vol. abs/1801.05927, 2018.

[26] Q. Yang, J. Suh, N. Chen, and G. Ramos, "Grounding interactive machine learning tool design in how non-experts actually build models." ACM, June 2018.

[27] M. Chen and H. Wang, "How personal experience and technical knowledge affect using conversational agents," in *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*, ser. IUI '18 Companion. New York, NY, USA: ACM, 2018, pp. 53:1–53:2.

[28] H. S. M. Cramer, V. Evers, V. S., M. W., and B. J. Wielinga, "Awareness, training and trust in interaction with adaptive spam filters," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 909–912.

[29] J. Escudero, E. Ifeachor, J. P. Zajicek, C. Green, J. Shearer, and S. Pearson, for the Alzheimer's Disease Neuroimaging Initiative, "Machine learning-based method for personalized and cost-effective detection of alzheimer's disease," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 1, pp. 164–168, Jan 2013.

[30] O. Rudovic, J. Lee, M. Dai, B. Schuller, and R. W. Picard, "Personalized machine learning for robot perception of affect and engagement in autism therapy," *Science Robotics*, vol. 3, no. 19, 2018.

[31] A. Kapoor, B. Lee, D. Tan, and E. Horvitz, "Interactive optimization for steering machine classification," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 1343–1352.

[32] O. Tamuz, C. Liu, S. Belongie, O. Shamir, and A. T. Kalai, "Adaptively learning the crowd kernel," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. USA: Omnipress, 2011, pp. 673–680.

[33] J. Cheng and M. S. Bernstein, "Flock: Hybrid crowd-machine learning classifiers," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW '15. New York, NY, USA: ACM, 2015, pp. 600–611. [Online]. Available: http://doi.acm.org/10.1145/2675133.2675214

[34] Z. He and G. G. Yen, "Comparison of visualization approaches in many-objective optimization," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, June 2017, pp. 357–363.

[35] R. Sahu and A. K. Chaturvedi, "Many-objective comparison of twelve grid scheduling heuristics," 2011.

[36] D. J. Walker, R. Everson, and J. E. Fieldsend, "Visualizing mutually nondominating solution sets in many-objective optimization," *Trans. Evol. Comp*, vol. 17, no. 2, pp. 165–184, Apr. 2013.

[37] Z. He and G. G. Yen, "Visualization and performance metric in many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 386–402, June 2016.

[38] L. Miqing, Z. Liangli, and X. Y., "How to read many-objective solution sets in parallel coordinates," *CoRR*, vol. abs/1705.00368, 2017.

[39] M. Li, S. Yang, and X. Liu, "Diversity comparison of pareto front approximations in many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2568–2584, Dec 2014.

[40] P. Lindroth, M. Patriksson, and A.-B. Strömberg, "Approximating the pareto optimal set using a reduced set of objective functions," *European Journal of Operational Research*, vol. 207, no. 3, pp. 1519 – 1534, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221710004868

[41] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[42] C. J. Petrie, T. A. Webster, and M. R. Cutkosky, "Using pareto optimality to coordinate distributed agents," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 9, no. 4, p. 269–281, 1995.

[43] K. Deb and D. K. Saxena, "On finding pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems."

[44] R. C. Purshouse and P. J. Fleming, "On the evolutionary optimization of many conflicting objectives," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 770–784, 2007.

[45] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, 2008.

[46] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn," in *ICML workshop on AutoML*, 2014.

[47] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[48] Nbconvert, "Nbconvert python package." [Online]. Available: https://nbconvert.readthedocs.io/en/latest/api/preprocessors.html?highlight=ExecutePreprocessor#nbconvert.preprocessors.ExecutePreprocessor

[49] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2962–2970. [Online]. Available: http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf

[50] L. Zimmer, M. Lindauer, and F. Hutter, "Auto-pytorch tabular: Multi-fidelity metalearning for efficient and robust autodl," *CoRR*, vol. abs/2006.13799, 2020. [Online]. Available: https://arxiv.org/abs/2006.13799

[51] "Cancer mortality rates for us counties," https://data.world/nrippner/ols-regression-challenge, accessed: 2020-16-07.

[52] "Bluejeans software," https://www.bluejeans.com/, accessed: 2020-10-10.

[53] "Ngrok service," https://ngrok.com/, accessed: 2020-10-10.

[54] "Imdb movies data," ftp://ftp.fu-berlin.de/pub/misc/movies/database/, accessed: 2019-09-15.

[55] T. S. F. C. Office, "Employee compensation data," https://data.sfgov.org/City-Management-and-Ethics/Employee-Compensation/88g8-5mnd, accessed : March 15, 2019.

[56] S. G. H. and L. E. S., "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Human Mental Workload*, ser. Advances in Psychology, P. A. H. and N. M., Eds. North-Holland, 1988, vol. 52, pp. 139 – 183.