QUESTO: Interactive Construction of Objective Functions for Classification Tasks

Subhajit Das[1] [iD], Shenyu Xu,[1] [iD] Michael Gleicher,[2] [iD] Remco Chang,[3] [iD] and Alex Endert[1] [iD]
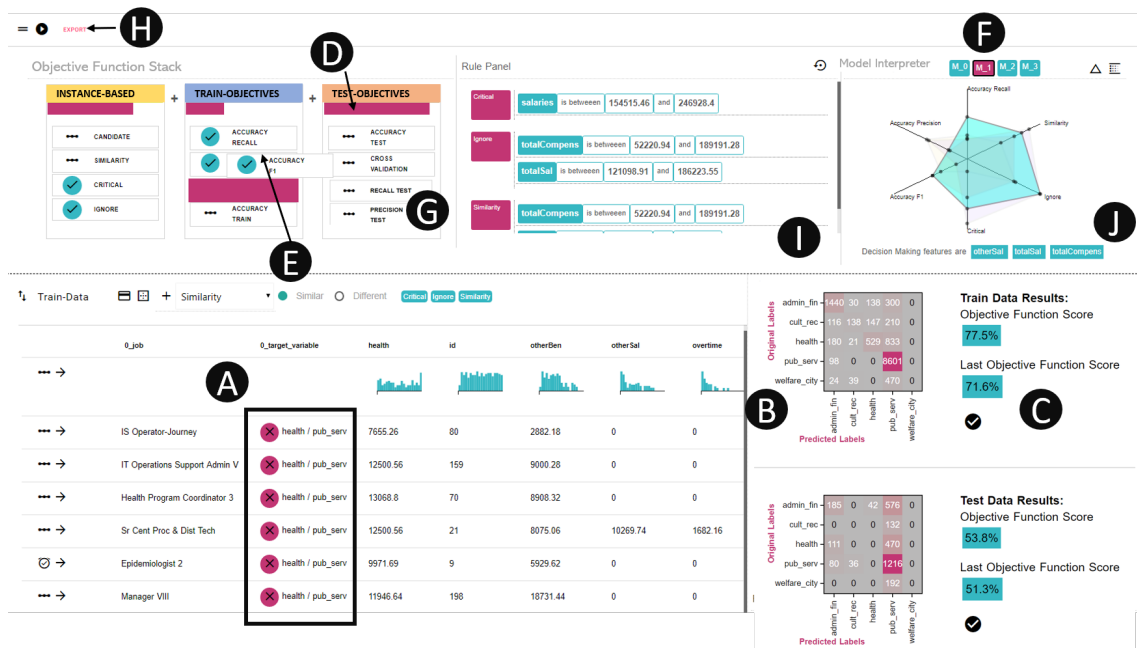
[1]Georgia Institute of Technology, USA   [2]University of Wisconsin – Madison, USA   [3]Tufts University, USA

Figure 1: *QUESTO: A. Incorrect predictions. B. Confusion matrices. C. Model score. D. Sliders to weight objectives. E. Draggable constraints to specify priority. F. Selected model. G. Test data constraints. H. Model run and Export button. I. Rule panel. J. Important features.*

Abstract

Building effective classifiers requires providing the modeling algorithms with information about the training data and modeling goals in order to create a model that makes proper tradeoffs. Machine learning algorithms allow for flexible specification of such meta-information through the design of the objective functions that they solve. However, such objective functions are hard for users to specify as they are a specific mathematical formulation of their intents. In this paper, we present an approach that allows users to generate objective functions for classification problems through an interactive visual interface. Our approach adopts a semantic interaction design in that user interactions over data elements in the visualization are translated into objective function terms. The generated objective functions are solved by a machine learning solver that provides candidate models, which can be inspected by the user, and used to suggest refinements to the specifications. We demonstrate a visual analytics system QUESTO for users to manipulate objective functions to define domain-specific constraints. Through a user study we show that QUESTO helps users create various objective functions that satisfy their goals.

CCS Concepts

● ***Computing methodologies*** → *Model construction and selection;* ● ***Mathematics of computing*** → *Interactive objective functions;* ● ***Human-centered computing*** → *Visual analytics;* ● ***Machine learning task*** → *Classification;*
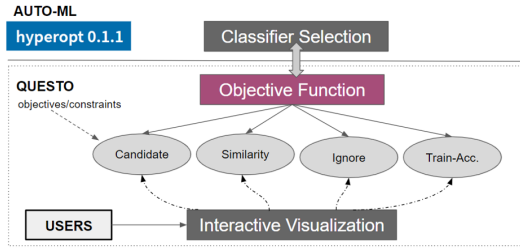
Figure 2: *QUESTO coupled with an Auto-ML optimizer - Hyperopt, translates user interactions into objective function terms.*

1. Introduction

Objective functions serve a crucial role in traditional machine learning (ML) tasks, such as classification. ML models are driven by the design of these objective functions which are solved to attain desired goals. They act as the mathematical expression of preferences, goals, and constraints that ML pipelines should take into consideration when learning from training data to create models. For example, in a classification task an objective function may specify that the model should perform well on chosen class labels or may require a model to get certain data instances (rows in the data) correct or may include regularizers to create generalisable models. ML practitioners often create and edit objective functions, specifying feedback to ML processes to achieve desired goals such as a model with a high test accuracy, or creating a fair classifier.

When objective functions are made interactive, they can serve as the medium through which people can directly communicate their domain expertise, preferences, and other relevant information to the system. However, specification of such intricate objective functions is difficult for users who do not have programming or ML expertise. For instance, creating custom objective functions require translating one's preferences, goals, and constraints into mathematical expressions by writing code. To support interaction with objective functions, we need interactive visual interfaces which can help users who seek to construct ML models, expressively communicate their preferences or intents to the underlying models. Even for expert ML users (who can write code), interactive construction of objective functions can help them rapidly test various forms of objectives to find optimal models that suits their goals.

There are numerous constraints or specifications that can be added to objective functions. For example, in a classification task a user might prefer models which correctly predict specific data instances [RSG16] (data instances are rows in a tabular data). Furthermore, users might expect to see similar data instances placed in the same class [KLTH10; TLB*11], remove data instances which are noise/outliers [TLKT09], etc. Based on a literature review of previous work focused on interactive construction of ML models (e.g., [ZSZR18; RSG16; YSCR18; KLTH10]), we investigated the design space of such constraints.

In this paper, we present a prototype visual analytic (VA) system called QUESTO that translates user interactions with data into objective functions for classification. QUESTO facilitates interactive construction of a classifier by allowing users to formulate their preferences as an objective function while they explore and interact

with a tabular dataset. QUESTO visualizes the underlying objective function to help users understand their specified goals and constraints. Furthermore, through iterative interactions with the data, the user can refine the specified constraints and the objective function to find a model that correctly characterises the data. The resulting user-defined multi-objective objective function can be used with Auto-ML optimizers (such as Hyperopt [KBE14]) to guide and facilitate the construction of classification models (see Figure 2), with the advantage of users being able to specify domain-specific constraints and objectives.

In addition, we present the findings of a within-subjects user study. In this study we compared QUESTO with traditional command line workflows (CMD) that allow construction of classifiers and objective functions by writing code, and with stand alone Auto-ML. Our study showed that: (1) QUESTO is easier and faster to use than CMD workflows to specify/test several objective functions to classifiers, and (2) Although QUESTO generates models which are slightly less accurate than models from stand alone Auto-ML, these models outperform Auto-ML in attaining user-specified constraints. Furthermore, we seek qualitative participant feedback to understand their response on QUESTO's interface/interaction design, and usability issues to further improve interactive construction of objective functions. Our contributions are:

- A design space of constraint specification derived from prior work which categorizes various user goals for a classifier.
- A prototype VA tool, QUESTO, for interactive objective function construction for classification tasks.
- A user study evaluating how people specify objective functions with QUESTO compared to writing them with code, as well as compared to existing Auto-ML approaches.

2. Related Work

2.1. Visual Analytics for Interactive Machine Learning

Many visual analytics systems incorporate demonstration-based interactions to adjust underlying models [BNHL14; EFN12; LHM*13; EHM*11; EBN13; GLG*13]. Demonstration-based interactions allow users to directly manipulate on-screen visual data representations to adjust models without the need to write code or adjust widgets in a control panel [BNHL14]. These visual interfaces have helped researchers discover novel workflows in interactive machine learning to assist users to build models aligned with their goals. For example, Gestalt [PBD*10] showed the integration of ML with traditional software development workflows. Their system implemented a classification model pipeline helping users in model construction and analysis. Other similar VA systems include [BGV16; RHY14; KPHH11; EFN12; SKBE17; KDJH08; KLTH10; TLKT09; SRL*07]. A complete overview of this interactive process is discussed by Amershi et al. [ACKK14].

VA systems use explicit user interactions to specify parameters of underlying models. These interaction techniques have proven useful in various domains of ML (other than classifiers [VDEW11]), such as clustering [WN17], dimensional reduction [KCPE16; KKW*17; EHM*11; JZF*09], metric learning [BLBC12], etc. Recent work utilised multiple ML models to satisfy user preferences [SHB*14; PSTW*16]. For example, Hypertuner

[TCWM18] and BEAMES [DCCE18] allowed tuning hyperparameters of multiple regression models to support user goals. Clustervision enabled exploring multiple cluster models to analyze the fitness based on metrics such as Silhouette Coefficient [KEV*18].

In general, these current VA tools interactively adjust ML models based on a pre-defined objective function (by ML practitioners or system developers) which users often cannot adjust or view. Such tools do not allow users to interactively design an objective function that seeks to adjust the behavior of ML models or select models that better suits the specified constraints. Rather, user interaction in these tools is interpreted as changing feature weights, or adjusting values of model parameters for model steering. Instead, this paper explores how people can interact with objective functions.

Past researchers have looked at human-in-the-loop based ML processes [AFKT11; BZSA17; BM17; KBS*11; GFT*16; KTD17] where they showed human contribution in ML processes in various ways. For example, the system Label-and-Learn facilitated a user-guided labeling process in ML using interactive visualization techniques [SLT17]. Squares showed humans interactively debugging ML models [RAL*17]. Liu et al showed an interactive system called CNNVis that helps ML experts debug, diagnose, and further improve CNN models [LSL*17]. Sacha et al. have examined various VA systems to discover that users evaluate models by a plethora of measures including conventional metrics such as, accuracy and cost, as well as novel criteria such as unexpectedness [SSZ*17]. Holzinger et al. have claimed that in cases of insufficient training data, ML problems are often NP-hard, which can be solved by interactive ML-based solutions using a domain-expert user's input [Hol16]. To increase human input, crowd-sourcing is a new avenue in human-machine collaboration [CFEC17]. Crowd workers perform various tasks such as, clean/transform data, anotate/add labels, etc. Novel techniques have improved the reliability of having crowd workers be part of conventional ML pipelines as shown in [CAK17; NKHK17; CB15; SRL*09; WLDW01].

2.2. Auto-ML Systems

Conventional ML pipeline requires selecting an appropriate learning algorithm, and finding an optimal combination of hyperparameters to construct a model. Evidently, this workflow is not suitable for novice ML users, who otherwise rely on GUI based tools such as WEKA [EW16]. To help novices construct better ML models, new tools are developed in the space of automated ML or Auto-ML such as Auto Weka [THHLB13; KTH*16], BigML [MLa] SigOpt [PPJ07], Hyperopt [BYC13; KBE14], Snowcat [CHH*19] and AUTO-SKLEARN [FKE*15]. Holzinger et al. explored Auto-ML based workflows in various domains to further confirm the usefulness of such systems [Hol16]. While useful, these tools are limited to optimize ML models based on conventional metrics such as *Precision*, or *Recall*. Users cannot specify constraints or metrics that are more meaningful/useful to them. In this paper, we seek to create a novel technique to incorporate domain expertise and user's subjective preferences in model construction, selection, and evaluation.

2.3. Visualizing Objective Functions and Solutions

Various techniques have been used to visualize solution sets from an objection function space such as MDS, RadViz, bubble chart, parallel coordinates, self organizing map, etc. [HY17]. Further, He et al. proposed a new visualization technique to map solutions from a high-dimensional objective space to a 2D polar coordinate plot. Their method helped a user to understand tradeoffs between objectives and find desirable solutions [HY16]. Sahu et al. showed the use of a radar chart to visualize many-objective solution spaces [SC11]. Walker et al. visualized a set of mutually non-dominating solutions using Radviz to show multi-objective solutions, and introduced techniques to measure the similarity of non-dominating solutions [WEF13; WFE12]. Many researchers have looked at measures to assess the diversity of pareto-optimal solutions in multi-objective optimization problems [MLY17; LYL14]. In QUESTO, we leverage these techniques to visualize solutions to a multi-objective objective function. However, the focus of our work is not on the visualization of the solution spaces, but in helping users express their analysis intent through objective functions.

2.4. User Preferences/Modeling

ML users solve various problems which are context dependent and personal [AF18], e.g, interaction with robots in a hospital [RLD*18; EIZ*13]. Diverse problem scenarios create an opportunity to specify a diverse set of user preferences. These preferences are the building blocks to construct an objective function. We studied the literature to understand what kind of specifications users can provide to construct a classifier [ZSZR18; YSCR18; CW18; CES*09; TDCF07]. Kapoor et al. discussed often users have to rely on the overall classification accuracy of predictive models instead of relying on predictions generated by marginal models. This often leads to a bad model selection [KLTH10]. Zhu et al. described the machine teaching paradigm where a machine teacher (usually a domain expert) shows informative data instances of positive and negative class labels to maximize the distance between the classes [ZSZR18].

Lime, a submodular optimization technique helped users interpret models by explaining the prediction of a classifier on a set of relevant data instances [RSG16]. This showed the relevance of a model's performance on a specific subset of data items that the users care about. Tamuz et al. showed an adaptive algorithm that estimated a similarity matrix from human judgments based on comparisons of triples [TLB*11]. Applying the same ideology in classification tasks, users can specify data instances that are similar and should be predicted to be in the same class label. The system Flock asked crowd workers to define the reason behind a pair of instances to be in a positive class and vice versa [CB15]. Kapoor et al. discussed if users can understand the model behavior, they can assess the possible next moves to adjust the model further [KLTH10]. For example, users can evaluate if models correctly predict similar data instances in the same class label. If these are not in the same class, users may provide additional examples to refine the model. These pioneer works in the literature inspired us to help users define their measure of success in a classifier construction process through interactive objective functions.

3. Design space of objectives and constraints

We define *objectives* as a component or a function term of a multi-objective objective function. These objectives are also referred to as user preferences, goals, subjective requirements, etc. in the paper. Next, each of these objectives may have *constraints*. For example a user may have an objective of finding a highly performant model. A constraint can be the accuracy be calculated using precision, recall, or F1-score metrics. Similarly another objective could be that the classifier performs better on hand-chosen data instances. In this objective one constraint may be that it labels similar data instances (specified by the user) in the same class label.

We analyzed 61 papers from visual analytics and interactive machine learning areas, which included 18 VA systems to formulate a set of objectives and their respective constraints for a classification task (†). Using an affinity diagramming approach, we clustered similar papers and VA systems with similar interactions. Further, we iteratively refined these until we were satisfied with the relationships between the clusters. For example, while *Critical* constraints address data instances that users find important to be correctly predicted, *Candidate* captures data instances that are strong representatives of a class label (and not necessarily *Critical*). Finally, we derived 15 constraints users can define, organized into 4 objective categories for a classification task: Instance-based, Feature-based, Train-objectives, and Test-objectives, described below.

3.1. Instance-based

This objective allows users to specify that the classifier should perform well on a set of data instances. While many of the constraints in this section involve adjusting weights on specified data instances, we categorized them separately based on how they are revealed in the user interface and the user task supported.

Similarity: This constraint captures the degree of similarity (or difference) between data items. A user specifies a set of similar data instances and expects them to be predicted in the same class label. Users can also specify pairs of data items to be predicted in different classes. However, in both cases ("similar" and "different") users do not specify the class in which the specified data instances should be placed. There are various VA systems/techniques where similarity and difference between data samples was sought from users to construct models. For example, Tamuz et al. discussed a similarity matrix to infer if an object "A" is similar to "B" or "C" for a user [TLB*11]. Another system, Flock, asked crowd workers to specify paired examples to define similar or different instances of positive or negative class [CB15].

Candidate: This constraint type refers to user feedback on a set of data instances from the training set which are good representatives of their class. Based on prior knowledge, users can specify that data instances represent a given class well.

Critical: This constraint lets users specify a set of data instances to show that correct prediction of these are critical for them. This constraint can be specified when users want a set of data instances to be correctly classified, while the accuracy of other data instances is

less important. For example, consider a financial analysis scenario where a company needs a classifier to predict which clients should be granted a loan. The analyst might know a few clients who are more profitable than others. Thus, he or she might prefer a classifier that correctly predicts these clients than the less profitable ones. Users can assess constructed classifiers based on how accurately they predict the specified critical data instances. Other researchers have looked at critical data instances. For example, Lime is a ML algorithm which helps users to interpret models by explaining their predictions on data instances that are critical [RSG16].

Ignore: This constraint specifies if the user intends to remove noisy or outlier data instances from the training set. Removal of noise or an outlier increases the accuracy of prediction and the power of the model to generalize on unseen data. This constraint may also include specification of data instances whose prediction (correct or incorrect) is irrelevant to users. Elzen et al. prototyped Baobab-View for inspecting outliers and noisy data to improve interactive construction of decision trees [VDEW11].

3.2. Feature-based

This category includes items which users can specify to help a model in its learning process. The following constraints defined under this category operates at the level of the features (or attributes).

Feature selection: This constraint allows users to specify features that are important for classification. A classifier will only use the specified features when it is learning the data or the Auto-ML system will penalize any model that uses features other than the ones specified by the user. This is inspired from previous work that showed the value of human-centered feature selection in model construction [KPB14; LWLZ17; CB15].

Correlation and Variance: This constraint allows users to specify correlation and variance in the features based on their domain expertise. Users specify perceived correlation between features and variance per feature in the data. *Correlation* and *Variance* are based on the user's domain knowledge and not necessarily grounded in the training data. For example, a financial analyst might know that experienced bank customers in the age group of $40-50$ are more likely to spend economically and pay on time. Thus the features *age* and *payment* are correlated based on the domain knowledge of the user, which may or may not be present in the data. Hall showed a feature selection method based on correlation in the data [H.99]. Instead of computing correlation in the features from the training data, users may specify correlation and variance in the data, an information that affects how the model learns from the data.

3.3. Train-objectives, and Test-objectives

ML models can be evaluated using conventional model metrics or constraints such as **Accuracy**, **Precision**, **Recall**, and **F1-Score**. When applied to the training set, we consider them *Train-objectives*, compared to applying them to the test set when we consider them *Test-objectives*. Together these two objectives help users control for model overfitting. Using the *Train-objectives* users can verify how well the model is learning from the data, while using the *Test-objectives* they can validate if the model generalises well

---

† The full list of papers is provided in the supplemental material

on unseen data. For example, one can first find models with high training accuracy (i.e., low bias), then tune the hyperparameters to achieve a high test set accuracy (i.e., low variance), and finally weight or rank these constraints to find classifiers that show an optimal tradeoff between bias and variance.

4. QUESTO: System Design and Description

QUESTO helps users interactively construct classifiers that address their subjective goals specified in the form of interactive objective functions. The interface is designed to facilitate interactive specification of constraints at the data instance and feature level. QUESTO enables users to see specified constraints, and add/remove/edit them. QUESTO supports classifier inspection and selection from a set of candidate models generated from the user-specified objective function with Hyperopt in the backend.

Upon data import, QUESTO splits the datasets into a training set, multiple test sets, and a final validation set. The training set is used to build the initial model. The multiple test sets are used for each iteration of user feedback to test how well the model meets the objectives and constraints specified. When users are satisfied, there is a final validation set (which is never seen by the model during the previous iterations) used to validate the model.

4.1. User Interface and Interactions

Data table view: It displays the training and test data sets in two data tables. In addition, it supports specification of 4 types of constraints: *Critical*, *Ignore*, *Similarity*, and *Candidate*. To specify *Critical* or *Ignore* constraints, users can click on the row (See Figure 4-E). To specify a *Similarity* constraint, users can specify either "Similar" or "Different" on pairs of data instances. To specify the *Candidate* constraint (see Figure 4-B, E), users can select rows within a class to specify example data instances. When users select a model from the model interpreter view, the class label column of both training and test data shows correct predictions with a check mark, while incorrect predictions are shown with a cross mark (see Figure 1-A).

Scatterplot matrix view: Shows the relationship between various attributes, and helps users find missing data, outliers, etc.

Confusion matrix: A confusion matrix for both the training and test dataset is shown on the right of the data table view (see Figure 1-C). Clicking on cells in the confusion matrix filters the data table to show data instances responsible for the prediction.

Feature panel: The features of the data are visualized as a parallel coordinate plot (Figure 4-G). Brushing on any feature (represented as vertical lines) allow users to filter the data instances in the data table. Users can specify a *Feature Selection* constraint by clicking on the cells over the vertical lines (Figure 4-F). Further, right clicking on these cells specifies a *Correlation* and *Variance* constraint.

Objective function stack: This view visualizes the interactive objective function. Each box (see Figure 1-G) represents an objective category and contains constraints under the categorization described in section 3. Users can specify a constraint by clicking on its name (a specified constraint is shown with a green checkmark, see Figure 1-E). When users explore and interact with the

data QUESTO automatically infers constraints such as *Candidate*, *Critical*, etc. However, users can override the inferred constraints if they disagree. Underneath each box is a slider (Figure 1-D, E), which allows users to adjust the weight of each objective. Users can also reorder constraints vertically to indicate priority.

Model interpreter: Shows metrics of k ML models. Users can choose to inspect each model by seeing it in a parallel coordinate or in a star plot (see Figure 1-F). Users can select a model which updates the data table view (showing predictions for each data instance) and updates the confusion matrices. This view also shows the top 3 features used by the classifier.

Rule Panel: When users specify any constraints by filtering data instances, QUESTO saves these data instances as part of a *rule* (Figure 1-I) that can be interacted with through this view.

4.2. Technique

This section describes the underlying techniques in QUESTO that drive the construction of an interactive objective function (see Figure 3). QUESTO uses an Auto-ML module called Hyperopt [KBE14] to find an optimal hyperparameter combination which may meet user goals. QUESTO specifies a learning algorithm, and a set of hyperparameters for Hyperopt to construct models (e.g., we used a random forest classifier with the hyperparameters *MaxDepth*, *Criteria* ("gini" or "entropy"), and *MinSamples*). Our technique uses an iterative model construction and evaluation workflow. Users formulate their preferences in the objective function O by specifying a set of constraints Φ, and a weight vector W. Per iteration Hyperopt consumes O to construct and rank new models M. This cycle allows users to search for models by defining their goals and constraints in O.

4.2.1. Classifier Creation and Selection

Model Construction: QUESTO splits the loaded data set Z, into training D, multiple test sets $T = T_1, T_2, T_3...T_t$, and a validation set V. Next it triggers Hyperopt to start the model optimization process on the supplied data D to build M classifiers of size N ($M = m_1, m_2, m_3, ...m_N$). It constructs each model m_i using a supplied learning algorithm κ and sampled B hyperparameter combinations. We used Scikit-Learn's machine learning package to build the classifiers [MLb] further supported by Hyperopt which provides new hyperparameter combinations. Finally, each of the trained model m_i predicts class labels for both D, and T_i. The models are evaluated based on their performance on the supplied objective function O.

Model Optimization: Hyperopt traverses the space of pre-defined hyperparameters to construct a set of models. As an input, Hyperopt expects a list of hyperparameters to tune, $H = h_1, h_2...h_B$ and a domain space ν_i for each hyperparameter h_i. We pre-selected B hyperparameters and their domain space. For example, if a domain space of $2 - 100$ for *MaxDepth* hyperparameter was chosen, Hyperopt will sample a value between 2 to 100. Further, using user-defined constraints such as *Testing-Accuracy* or *Cross-val-score*, Hyperopt samples the hyperparameter space to control for model overfitting. Hyperopt evaluates each model in M with the supplied objective function O. The objective function O is constructed by a weighted linear combination of user-defined
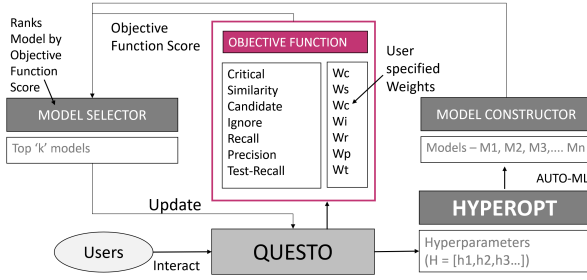
Figure 3: *QUESTO system architecture.*

constraints Φ. Each model m_i in M gets a objective function score $S = s_1, s_2, ...s_N$ based on its performance on O. Hyperopt ranks the N classifiers (we set $N = 100$, but can be changed) based on the objective function scores in S.

4.2.2. Interactive Objective Function Creation

The set of constraints Φ represent the different categories of goals and constraints in our taxonomy (see Section 3). Using QUESTO, users can define the following constraints interactively:

Critical: Hyperopt trains a model m_i on the supplied training set D with original labels $L_{d,o}$. Next it predicts labels on the training set as $L_{d,p}$. Users specify a list of IDs of x critical data instances $C = c_1, c_2...c_x$. QUESTO retrieves the prediction on the critical data instances based on supplied IDs C as $L_{c,p}$ ($L_{c,p} \subset L_{d,p}$). QUESTO retrieves the original class labels of the critical data instances as $L_{c,o}$. QUESTO initializes the score for the *Critical* constraint as $s_1 = 0$. It compares $L_{c,o}$ with $L_{c,p}$, to find the number of correct matches and save in s_1 (normalized between 0 to 1).

Similarity: This constraint is of two types. The first type captures similar data items when a user specifies a list of data items Y that are similar and should be placed in the same class. The algorithm iterates over Y to find the original class label $L_{n,o}$. If Y belongs to more than one class label, the most frequent class label is assigned to $L_{n,o}$. Next the algorithm matches the prediction $L_{n,p}$ with the original class label $L_{n,o}$, where i is the index, iterating from 0 to b (the number of user-specified similar data instances). For every correct match, the score for this constraint $s_{2,a}$ gets a +1 score. Next, it normalizes $s_{2,a}$ as $(s_{2,a})/b$, to get a score between 0 to 1.

The second type consists of users specifying items that should be different. A user does so by specifying a list of tuples σ, where each item in the list is represented as $\sigma_i = (\sigma_x, \sigma_y)$. Here σ_x and σ_y are data instances that users expect to be predicted in different classes. The algorithm iterates over σ and matches the predicted class label $L_{\sigma,p,i}$ of item σ_i to class label $L_{\sigma,p,j}$ of item σ_j. For every incorrect match, the score for this constraint $s_{2,b}$ gets a +1 point. Next, it normalizes $s_{2,b}$ as $(s_{2,b})/b$, to get a score between 0 to 1. Finally the score for this constraint is computed as $s_2 = (s_{2,a} + s_{2,b})$. We normalize s_2 to get a score between 0 to 1.

Candidate: Users specify a list of b data items in G with a class label A. The algorithm increases the training data weights ($W = w_1, w_{s2}, w_{s3}...w_b$) for these data items. It inputs W to Hyperopt which trains the classifiers M based on the updated weights. The algorithm iterates through G and matches each items predicted class

label $L_{f,p}$ with A. The score for this metric s_3 is initialized as 0. For each correct match in the iteration, s_3 is assigned a +1 point. Finally the algorithm normalizes s_3 as $s_3 = s_3/b$.

Ignore: Users specify a list of data items I. The algorithm removes these data items from D to form a new training set D_{II}. Further when computing classification model metrics such as *Precision*, *Recall*, etc. the algorithm removes the data items in I from D.

Accuracies: This captures the classification model metrics defined as part of the category *Train-objectives* and *Test-objectives* (see section 3). These are *Precision*, *Recall*, *F1-Score*, etc. For each of these, QUESTO uses Scikit-Learns classification metrics [MLb].

Feature Selection, Correlation, and Variance: QUESTO incorporates a user-based feature selection technique. Users can guide feature selection in one of two ways. First, users can select a set of features F_d using the feature panel. QUESTO interpretes that the user chose a set of features which has a high correlation with the class label. For model construction QUESTO will only use the features in F_d. Second, users can specify a set of correlated features F_c or features with low variance in F_v. QUESTO automatically discards the features which show high correlation (negative or positive) with each other and selects the ones which are uncorrelated. Similarly QUESTO discards features with low variance. The selected features $F = F - F_c$ or $F = F - F_v$ are fed to Hyperopt.

Objective Function Formulation: The resulting objective function in QUESTO is a weighted linear combination of constraints Φ. The algorithm computes the scores for each constraint. Finally, it linearly combines these individual constraint scores as shown in the equation: $O = s_1 * \omega_1 + s_2 * \omega_2 + s_3 * \omega_3 + s_4 * \omega_4 + s_5 * \omega_5$ where s_1, s_2, s_3, etc. are the scores of the user-specified constraints.

Weighting Preferences: Initially the weights $W = \omega_1, \omega_2, ...\omega_U$ are evenly set for each constraint in Φ However, users can override and specify weightings for each constraint by interaction techniques shown in Section 4.1. The weight vector W sums to 1, to ensure correctness in computing the contribution of each item in Φ.

Guarding Against Data Leakage: QUESTO searches for best models from a pool of sampled ML models (facilitated by Hyperopt). This is different from optimizing the internal algorithm of ML models such as designing the loss function for a random forest. As our technique treats each model as a black box, it can optimize for models that may perform better on train or test data. This is similar to conventional model construction processes, where ML practitioners aim for a higher training or test accuracy or a balance between each. However, to guard against data leakage, our technique uses multiple sets of test data $T = T_1, T_2...T_t$ (each used per iteration of model construction). When QUESTO loads input data Z it automatically splits into three sets: train D, test T, and validation V. Next it makes t subsets of the test data, each used per iteration of model construction. This serves to address potential data leakage as the models are tested on unseen data and during training, the test data is not used. Per iteration, the model gets a new test set T_i to compute the objective score (test-accuracy, test-F1-score etc.). In addition, when the model is exported by the user they get to validate it on the validation set V, which is not used on any prior iteration.

5. Usage Scenario

We describe a scenario showing how QUESTO can help users build classifiers on an imbalanced dataset through the specification of interactive objective functions. Chris is a public policy analyst (with basic knowledge of ML), is analyzing San Francisco's employment data to predict employees department [Off] (10000 samples). Chris explores the salary data in QUESTO and begins constructing classifiers. They inspect the best classifier from the model interpreter to observe that it scored poorly on both training and test set (avg. 48% accuracy). Chris explores the confusion matrix to discover that 80% of the original labels are of type *public service* indicating that the data is imbalanced (there are 5 class labels). While the current model correctly predicted 70% of this label type, it failed to predict the other labels accurately. Chris expects the classifier to predict each class label with at least 80% accuracy to be valuable for their analysis. Next, using the objective function stack view Chris specifies the constraint *F1-Score* to account for the imbalanced nature of the dataset, and the constraint *Cross-val-score* to construct generalisable models.

Chris triggers QUESTO to construct a set of new classifiers (seen through the model interpreter) based on the updated objective function. They click on model 2 as it shows the best performance on all the specified constraints in the star plot view. However, other than the class *public service*), this model failed to perform as per expectation on other labels. For example, the label *cultural/recreation* and *admin/finance* showed 0 correctly predicted data items. Motivated to resolve these two classes, Chris brushes on the feature *retirement* on the feature panel to filter jobs with high *retirement* benefits. Next, Chris picks the constraint *Candidate* from the drop-down menu as seen in Figure 4-B. They click on a set of data items of label *administration/finance* on the table (shown with a green highlight, see Figure 4-E). When Chris specifies the *Candidate* constraint, QUESTO automatically records the selected data instances as part of a *rule* (see Figure 1-I). Chris assigns it a custom name *high-retirement*. Next, they filter the data instances using the feature panel with low *overtime* and high *salary* values. Chris specifies a subset of these data instances as *Candidate* constraint for the class *cultural/recreation*. QUESTO saves these data instances as a *rule*, which Chris calls *high-sal* (Figure 1-I).

Chris inspects one of the new models (model 2) by hovering over the two rules (*high-retirement* and *high-sal*, shows associated data items in the table view). Quickly glancing through these data items, Chris observes that the prediction error is less than 10% which they find acceptable. Next, Chris explores the confusion matrices and finds that the performance score for the model increased to 72% accuracy (see Figure 5-B,C). Furthermore, Chris notices that the model's performance improved significantly achieving on average 70% accuracy for all classes except *admin/finance*. In order to boost the performance of the label *admin/finance*, Chris specifies all the data instances of this label (in the training set with more than 80000 *retirement benefits*, total 12 items) as a *Critical* constraint.

Iterating further Chris finds a new set of classifiers from the collection satisfied the constraints *Candidate*, *Critical*, and *F1-Score*. However, the performance dropped on the test set in all of these models. To further emphasize the performance on the test set as an important criteria, Chris specifies the *Testing-Accuracy* constraint.
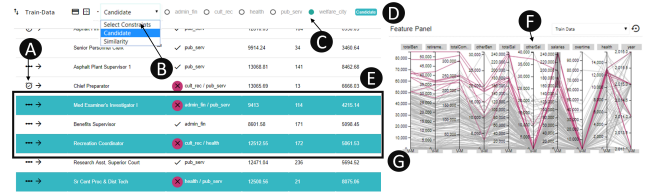

Figure 4: *A. Critical constraint. B. Other constraints. C. Class selector. D. Filter tags. E. Selected data. F. Select features, correlation, and variance. G. Parallel coordinate plot.*
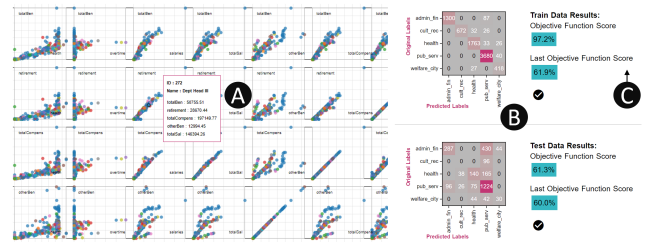

Figure 5: *Views for Evaluating Model Performance.*

In addition, they lower the weights on the constraints in the categories *Train-objectives* and *Instance-based*, and increases the weight on the category *Test-objectives* (see Figure 1-D, E). They also re-order individual constraints specifying highest priority to the constraint *Candidate*. Chris constructs new classifiers and chooses model 3 based on its performance as viewed in the star plot view. They hover over the two custom rules (see Figure 1-I) to see that most of the data instances in them are correctly predicted and performs equally well on all class labels. Content with the current model, Chris tests the model on a validation set and then exports it for future use. This usage scenario shows how QUESTO enabled a domain expert to build an accurate classifier on an imbalanced dataset by exploring various objective functions that captured his subjective preferences.

6. Evaluation

Other than using interactive visual interfaces like QUESTO, two popular alternatives to constructing classifiers are: (1) manually creating them via code or command-line (CMD) interfaces, and (2) automatically generating them using Auto-ML. We wanted to compare QUESTO with manually coding to verify if QUESTO is easier and faster to specify constraints. Furthermore, we wanted to compare QUESTO with Auto-ML to verify if QUESTO satisfies subjective user goals better, and to compare the resulting accuracy. We conducted a within-subject controlled lab user study of QUESTO comparing it with CMD, and Auto-ML to construct a classifier. Our study addresses the following research questions:

RQ1 Is QUESTO easier and faster to use than CMD workflows?
RQ2 Does QUESTO build more accurate models than automatically-generated models from Auto-ML?
RQ3 Does QUESTO build models that addresses user-specified constraints better than models from Auto-ML?

We hypothesize the interactive visual interface of QUESTO will

be easier and faster to use compared to command-line interfaces. Here "faster" refers to the time it takes to specify constraints and construct classifiers; it does not include the time needed to train classifiers. Further, we hypothesize that Auto-ML will generate more globally accurate models, given the objective function optimizing towards a specified accuracy metric such as high cross-validation score. However, **RQ3** tests our hypothesis that QUESTO will be better at building models that fit specific user constraints given the customized objective function, thus illuminating this tradeoff between accuracy and user goals.

In the literature there is no well-defined metric to measure how well user-defined constraints are satisfied in classifier construction. Thus, we defined **constraint satisfaction score** as a metric that captures how well a model attains user-specifed constraints such as *Critical*, *Candidate*, etc. These scores are normalized between $0 - 1$ (higher is better). It is expressed as $(\sum_0^U \omega_i * \gamma_i)/U$, where ω is the weight of U constraints, γ is the score of each constraint, U is the total number of specified constraints in the objective function.

We recruited 16 participants (7 Male, 9 Female), aged between 21-29 (M=25, SD=2.91), by inviting participants through our university mailing lists. We required them to at least have a basic expertise in writing python code with elementary understanding of ML. All of the participants (undergraduate and graduate students) were familiar with basic/intermediate data analysis using tools such as MS Excel, Tableau, etc. The experiment lasted 60 minutes and we compensated each participants with a $10 Amazon gift card. The study was conducted using a 17-inch display and a mouse.

6.1. Study Design

We began the study with a practice session teaching users how to interact with both QUESTO and CMD (with VS Code as the scripting editor). During the practice session, participants were encouraged to ask questions to clarify uncertainty in relation to the workflow or the interaction design. We proceeded to the experimental sessions only when the participants felt confident in using each system. For quantitative evaluation each participant interacted with both QUESTO and CMD using one dataset. Furthermore, the order of the interfaces (QUESTO and CMD), the datasets, and the tasks were randomized to remove any ordering/learning effect. We considered these dependent variables for quantitative analysis: (1) *Task completition times:* in specifying an objective function through QUESTO or CMD, (2) *Model accuracy:* the accuracy of the model constructed using QUESTO, CMD, and Auto-ML stand alone, (3) *Constraint satisfaction score:* measure to capture constraints satisfied by the model, and (4) *User preference rating:* average preference rating of *Ease of Use* for QUESTO and CMD.

6.2. Datasets

For the practice session, we provided a dataset of 10000 credit card transaction records [YL09]. The data has attributes such as *bill paid month 1*, *bill paid month 2*, *bill paid month 3*, *account balance*, etc. It was a binary classification task to predict if a bank customer will default on a bank loan or not. For the experimental sessions, we provided San Francisco's employment dataset [Off] containing 5000 data items for the quantitative evaluation. Each row in the data

contains information about an employee's remuneration containing attributes such as *dental benefits*, *annual salary*, *monthly salary*, *extra benefits*, etc. The task was to predict the employee's department (5 classes). For the qualitative evaluation we used a movies dataset [Mov] (5000 samples) containing attributes such as *budget*, *gross revenue*, *facebook likes of lead actors, director, cast*, etc. The data has three labels for movie rating: high, medium, and low. Following best ML practices, we use multiple test datasets, so that the constructed model never sees unseen data instances,

6.3. Tasks and Procedure

In the practice session, we asked users to perform 2 tasks per interface. The first task was to design an objective function in QUESTO by specifying a *Critical* constraint. They were asked to iterate multiple times and improve the classifiers performance by refining the constraint. The next task was to add two more constraints (*Similarity*, and *Candidate*) to the objective function in QUESTO and iteratively improve the classifiers performance. They repeat the same tasks by writing code and using CMD to run their code to construct a classifier. For the quantitative evaluation participants are randomly assigned an interface (QUESTO or CMD). Participants were encouraged to think aloud while they interact with each system. The interviewer was a silent observer and was away from the participant to mitigate *Hawthorne and Rosenthal* effect during the session. Participants performed 3 tasks per interface (6 tasks total). Each of the tasks were in increasing level of difficulty.

Task 1 Specify the constraint *Similarity*. (Max time: 3 minutes)
Task 2 Specify *Critical*, and *Candidate*. (Max time: 5 minutes)
**Task 3** Specify *Critical*, *Similarity*, *Candidate*, *F1-Score*, *Precision*, *Accuracy*, and *Cross-Validation*. (Max time: 7 minutes)

For the qualitative evaluation, we asked participants to freely use QUESTO (may specify any constraint) to build a classifier (on the movies data), and improve its objective function score in 5 minutes.

6.4. Data Collection

For quantitative assessment, we saved log data which stores (per iteration) models selected by users, their learning algorithms, and hyperparameters, predicted class labels, interacted data items, user-specified constraints etc. When participants completed the three tasks on both interfaces, we asked them to fill a NASA-TLX form [HS88], and a post-study questionnaire with a set of likert scale questions. We asked questions to seek user preference rating to each interface's various dimensions such as: (1) Ease of use, (2) Flexibility, (3) Fun to use, (4) Learnability, and (5) Intuitiveness. After the qualitative evaluation we conducted a semi-structured interview asking open-ended questions about the workflow, system usability, and interaction design for each interface. In this interview we asked questions such as: (1) *Describe your thought process while you interacted with QUESTO?*, (2) *Explain your experience in constructing an objective function interactively vs, through writing code?*, (3) *How do you think we can improve the current workflow, and design of QUESTO?* We also captured video and audio of participants screen while they interacted with QUESTO.
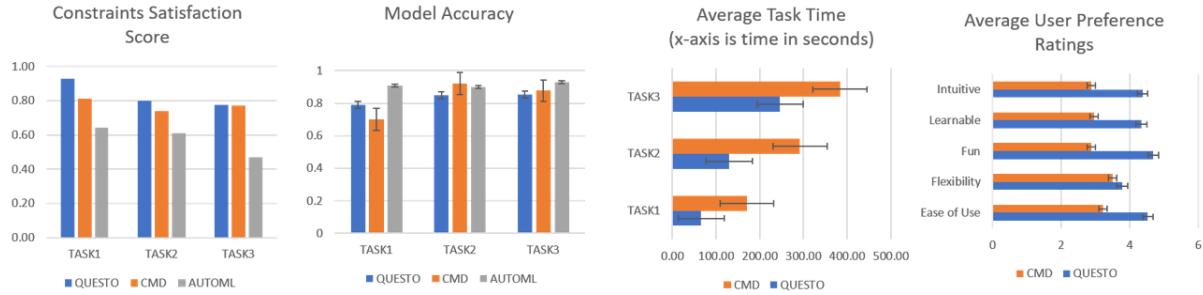
Figure 6: *The study results comparing QUESTO with coding interfaces and Auto-ML techniques for classification tasks.*

6.5. Quantitative Analysis

For the following we used the Friedman Test for Repeated-Measures as it is a good indicator of statistical significance for multi-class classifiers with multiple datasets, which may not follow a normal distribution as suggested by [Dem06]. Furthermore, we utilised Post-hoc Wilcoxon signed-rank tests with Bonferroni correction (new p value = 0.03) to test for statistical significance.

Ease of use: To answer **RQ1** we used likert scale rating scores (5-point scale) from participants (average ease of use rating 4.64, higher is better, see Figure 6-D). The ease of use of QUESTO was significant across all tasks: ($\chi^2(1) = 48.03, p < 0.03$). thus answering **RQ1** that QUESTO is easier to use than CMD.

Task completion times: Participants were significantly faster to specify constraints in QUESTO than CMD (Figure 6-C). Every participant completed all tasks except for three who failed to complete task 3 in the allotted time using CMD. In these cases, we recorded the maximum alotted time for that task. Quantitatively we found statistically significant difference in task completion time for all tasks: Task1 ($\chi^2(1) = 16.0, p < 0.03$), Task2 ($\chi^2(1) = 16.0, p < 0.03$), and Task3 ($\chi^2(1) = 4.0, p < 0.03$). This answered **RQ1** that QUESTO is faster to specify constraints than CMD.

Model Accuracies: To answer **RQ2**, we compared the models generated using QUESTO, CMD, and Auto-ML with respect to their accuracies. We found QUESTO generated similar quality models in relation to accuracies as produced by CMD (refer Figure 6-A). However, accuracies generated by models from Auto-ML stand alone were higher in comparison to QUESTO. We observed statistically significant difference in the model accuracy for task 1 ($\chi^2(1) = 12.25, p < 0.03$), and task 3 ($\chi^2(1) = 4.0, p < 0.03$).

Constraint satisfaction: We found that QUESTO outperformed Auto-ML in meeting user-specified constraints for all tasks: Task1 ($\chi^2(1) = 6.25, p < 0.03$), Task2 ($\chi^2(1) = 2.25, p < 0.03$), and Task3 ($\chi^2(1) = 9.0, p < 0.03$). This answered **RQ3**. We analysed this based on data collected from the employment dataset [Off] (users were given the example data items to specify as constraints), and on the movie dataset [Mov] (users freely specified constraints).

6.6. Qualitative Analysis

User Preferences: We measured user preferences using 5-point likert scale rating. QUESTO's user preference ratings (out of 5) were higher than CMD workflow in various aspects such as, *Easy of use* (Q: 4.53, CMD: 3.22), *Fun to use* (Q: 4.69, CMD: 2.88) , *Learnable* (Q: 4.34, CMD: 2.97), and *Intuitive* (Q: 4.38, CMD: 2.87) (see Figure 6-D). In addition, through the NASA-TLX survey we found that average participants' cognitive workload was significantly lower in QUESTO than CMD interfaces (Q: 4.30, CMD: 8.87, out of 10; lower is better). Based on these results it is likely that participants preferred QUESTO compared to using the CMD interface for objective function creation. However, many participants found CMD comparably flexible to QUESTO to specify preferences (Q: 3.78, CMD: 3.52) and debug models.

Easy workflow: Every participant found QUESTO's interface and the workflow easy to learn and use. Users liked the table view, with the ability to filter and search for specific data items as examples for constraint specification. P03 said, *"I like how I can mouseover on the cells of the confusion matrix to see incorrectly predicted data items."* P08 added, *"I frequently brush over the feature panel to filter data items by a set of attributes I care about."* However, users recommended that more advanced sorting feature in the table would have helped them find appropriate examples quickly.

Level of Detail: Some participants had elementary knowledge of python and ML, while a few others had in-depth ML expertise. P09 reflected *"Best part of this workflow is how easy the tool makes to change my constraints in terms of weighting or ordering them, which lets me look into a new set of classifiers"*. While advanced ML users appreciated the idea of an interactive interface to define an objective function, reflected that they would prefer to look at the numerical weights on each constraint so that they can better trust how the modeling engine is selecting classifiers. P13 said *"Though the objective function view is very useful, I am not sure if I understand what's happening on the background without knowing the numerical weights on these constraints.*

Meaningful constraints: Participants found the constraints in QUESTO appropriate and useful for classifier construction. P10 added *"I think the constraints make a lot of sense to me. When I test models I frequently look at specific data items to verify if the classifier modeled the data correctly"*. However, a few participants commented that it would help if QUESTO could recommend items to consider next. For example, P03 suggested *"I think you can use prediction probabilities to reflect on how confident the model is on each prediction. That may help me specify better examples."*

Intuitive model selection: Users found the model interpreter view

useful to compare models based on their performance on the specified constraints. P17 noted *"From the star plot view, I can inspect the size of the polygons to select models that performed better on relevant constraints. I would prefer to mouseover (instead of click) to browse the model output results on the table and the confusion matrices."* However, participants wanted to see a bookmark feature to save models they like. Furthermore, one participant desired to see how each constraint contributed to a change in model output to understand what interaction improved performance.

Task complexity: The three tasks in the user study had different levels of complexity. While task 1 and task 2 involved satisfying only subjective user preferences, task 3 included finding models that are accurate yet address constraints such as *Similarity*, *Candidate*, etc. Participants found task 3 more challenging, as finding the right model that is accurate and addresses their personal goals was difficult. P15 noted *"It was hard to improve the classifiers performance when I had more than 4 constraints to specify. However, with QUESTO I could rapidly test different weightings/rankings to these constraints to find an optimal classifier"*.

Conflicting constraints: We observed that participants sometimes specified conflicting constraints. For example, they specified a set of data item as a *Candidate* constraint, but in a later iteration they specified a subset of these data items as a *Ignore* constraint. As QUESTO currently does not support alerting users about conflicts, in future we plan to mitigate them in objective specification.

Modeling process: From the study we found that QUESTO's workflow does not help users understand the underlying modeling process. Often participants wondered what interaction in the previous iteration led to the improvement or degradation in the accuracy or the objective function score. Future work could make the modeling process more transparent and allow users to reason about the impact of their objective specification on classifier selection.

7. Discussion

User-System Feedback loop: Interactive objective functions may guide an Auto-ML model solver to select models that weigh data instances preferentially to better support user goals. For example, an objective function with the *Critical* constraint will guide the Auto-ML solver to prefer classifiers that correctly predict specific items. In response, users may inspect a model in relation to how well it supports the specified constraints. Thus interactive objective functions employ a two-way feedback loop between the user and the system for: (1) communicating preferences to find suitable models, and (2) providing a medium for understanding classifier performance. Participants confirmed that (per iteration) they validated models in relation to how well they support the constraints.

Tradeoff Analysis: Practically, in a multi-objective optimization problem, satisfying every objective might not be feasible [SAS02] due to a plethora of reasons including conflicting constraints, mathematical infeasibility, noise/outliers in the data, high dimensionality, etc. [KLTH10]. Formalizing user goals as a set of constraints in an objective function facilitates searching for a set of pareto-optimal solutions which may only satisfy a subset of the specified user goals. Furthermore, visualizing these pareto-optimal models help users inspect them and understand tradeoffs.

ML users often seek models that support other customized subjective goals/objectives that are personal and problem-specific ([KLTH10]). This study validated that while QUESTO produced slightly less accurate models than Auto-ML model solvers, they met subjective user-specified constraints. We envision that users will need to find a balance between these two extremes.

7.1. Limitations

Scope and assumptions of the user study: In the study we timed each task to ensure the study can be completed within a reasonable time. However, in more realistic settings, optimizing for (or limiting) time may not be meaningful. Further, we used pre-defined hyperameters within which Hyperopt sampled values to construct classifiers. The study results may vary if the set of hyperparameters used are different. Also, our study only included participants with basic expertise of python and ML.

Model overfitting: Good ML practices entail that trained models have no knowledge of test data. In QUESTO we follow the same process. The test data view in the data table allows users to inspect classifiers by reviewing the predictions made at the data instance level. Furthermore, by specifying the objectives *Train-objectives*, and (and weighting them), users can control for model overfitting. Nevertheless, we are aware that if users do not specify , they may produce overfitted models. However, this may be the case for command-line classifier construction as well if addition of regularizers or cross-validation approaces are not used.

Scalability: The current UI design is based on datasets of modest size. Thus, there are aspects of the UI that would become less usable if datasets grew larger. For example, while the table view supports sorting and filtering the fundamental design of showing items in a table may make finding individual relevant data items challenging. However, for larger datsets we can augment QUESTO with visualisations that can aggregate data and show example data items on demand (e.g., grouped heatmap view, etc.).

8. Conclusion

In this paper we present a design space of constraints, categorizing various user goals for a classification task. Grounded in this design space, we developed a prototype system QUESTO that supports interactive construct of objective functions to specify their subjective preferences to a classifier. Finally we validate the effectiveness of QUESTO by a within-subject user study that proved that QUESTO is easier and faster to use than conventional command line alternatives. We also demonstrate that while QUESTO produces slightly less accurate classifiers than Auto-ML, it produces models that better satisfy user-specified constraints. This presents a step towards understanding how to integrate people into Auto-ML processes.

9. Acknowledgements

References

[ACKK14] AMERSHI, S., CAKMAK, M., KNOX, W. B., and KULESZA, T. "Power to the people: The role of humans in interactive machine learning". *AI Magazine* 35.4 (2014), 105–120 2.

[AF18] AHAMED, F. and FARID, F. "Applying Internet of Things and Machine-Learning for Personalized Healthcare: Issues and Challenges". *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*. 2018, 19–21. DOI: 10.1109/iCMLDE.2018.00014 3.

[AFKT11] AMERSHI, S., FOGARTY, J., KAPOOR, A., and TAN, D. S. "Effective End-User Interaction with Machine Learning." *AAAI*. 2011 3.

[BGV16] BOUALI, F., GUETTALA, A., and VENTURINI, G. "VizAssist: An Interactive User Assistant for Visual Data Mining". *Vis. Comput.* 32.11 (Nov. 2016), 1447–1463. ISSN: 0178-2789. DOI: 10.1007/s00371-015-1132-9. URL: http://dx.doi.org/10.1007/s00371-015-1132-9 2.

[BLBC12] BROWN, E. T., LIU, J., BRODLEY, C. E., and CHANG, R. "Dis-function: Learning distance functions interactively". *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 2012, 83–92. DOI: 10.1109/VAST.2012.6400486 2.

[BM17] BIRAN, O. and MCKEOWN, K. "Human-centric Justification of Machine Learning Predictions". *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI'17. Melbourne, Australia: AAAI Press, 2017, 1461–1467. ISBN: 978-0-9992411-0-3 3.

[BNHL14] BRADEL, L., NORTH, C., HOUSE, L., and LEMAN, S. "Multimodel semantic interaction for text analytics". *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 2014, 163–172. DOI: 10.1109/VAST.2014.7042492 2.

[BYC13] BERGSTRA, J., YAMINS, D., and COX, D. D. "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms". *Proceedings of the 12th Python in Science Conference*. 2013, 13–20 3.

[BZSA17] BERNARD, J., ZEPPELZAUER, M., SEDLMAIR, M., and AIGNER, W. "A Unified Process for Visual-Interactive Labeling". *EuroVis Workshop on Visual Analytics (EuroVA)*. Ed. by S., MICHAEL and T., CHRISTIAN. The Eurographics Association, 2017. ISBN: 978-3-03868-042-0. DOI: 10.2312/eurova.20171123 3.

[CAK17] CHANG, J. C., AMERSHI, S., and KAMAR, E. "Revolt: Collaborative Crowdsourcing for Labeling Machine Learning Datasets". *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, 2017, 2334–2346. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3026044. URL: http://doi.acm.org/10.1145/3025453.3026044 3.

[CB15] CHENG, J. and BERNSTEIN, M. S. "Flock: Hybrid Crowd-Machine Learning Classifiers". *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. CSCW '15. Vancouver, BC, Canada: ACM, 2015, 600–611. ISBN: 978-1-4503-2922-4. DOI: 10.1145/2675133.2675214 3, 4.

[CES*09] CRAMER, H. S. M., EVERS, V., S., VAN, et al. "Awareness, Training and Trust in Interaction with Adaptive Spam Filters". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, 2009, 909–912. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518839 3.

[CFEC17] CROUSER, R. J., FRANKLIN, L., ENDERT, A., and COOK, K. "Toward theoretical techniques for measuring the use of human effort in visual analytic systems". *IEEE transactions on visualization and computer graphics* 23.1 (2017), 121–130 3.

[CHH*19] CASHMAN, D., HUMAYOUN, S. F., HEIMERL, F., et al. "A User-based Visual Analytics Workflow for Exploratory Model Analysis". *Computer Graphics Forum* 38.3 (2019), 185–199. DOI: 10.1111/cgf.13681 3.

[CW18] CHEN, M. and WANG, H. "How Personal Experience and Technical Knowledge Affect Using Conversational Agents". *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*. IUI '18 Companion. Tokyo, Japan: ACM, 2018, 53:1–53:2. ISBN: 978-1-4503-5571-1. DOI: 10.1145/3180308.3180362 3.

[DCCE18] DAS, S., CASHMAN, D., CHANG, R., and ENDERT, A. "BEAMES: Interactive Multi-Model Steering, Selection, and Inspection for Regression Tasks". *Symposium on Visualization in Data Science(VDS)*. Berlin, Germany, 2018 3.

[Dem06] DEMŠAR, JANEZ. "Statistical Comparisons of Classifiers over Multiple Data Sets". *J. Mach. Learn. Res.* 7 (Dec. 2006), 1–30. ISSN: 1532-4435 9.

[EBN13] ENDERT, ALEX, BRADEL, LAUREN, and NORTH, CHRIS. "Beyond Control Panels: Direct Manipulation for Visual Analytics". *IEEE Computer Graphics and Applications* 33.4 (2013), 6–13. ISSN: 0272-1716. DOI: 10.1109/MCG.2013.53 2.

[EFN12] ENDERT, A., FIAUX, P., and NORTH, C. "Semantic Interaction for Visual Text Analytics". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12. Austin, Texas, USA: ACM, 2012, 473–482. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2207741 2.

[EHM*11] ENDERT, A., HAN, CHAO, MAITI, D., et al. "Observation-level Interaction with Statistical Models for Visual Analytics". *IEEE VAST*. 2011, 121–130 2.

[EIZ*13] ESCUDERO, J., IFEACHOR, E., ZAJICEK, J. P., et al. "Machine Learning-Based Method for Personalized and Cost-Effective Detection of Alzheimer's Disease". *IEEE Transactions on Biomedical Engineering* 60.1 (2013), 164–168. ISSN: 0018-9294. DOI: 10.1109/TBME.2012.2212278 3.

[EW16] EIBE, F. MARK A. H. and W., IAN H. *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. 2016 3.

[FKE*15] FEURER, M., KLEIN, A., EGGENSPERGER, K., et al. "Efficient and robust automated machine learning". *Advances in Neural Information Processing Systems*. 2015, 2962–2970 3.

[GFT*16] GILLIES, M., FIEBRINK, R., TANAKA, A., et al. "Human-Centred Machine Learning". *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. CHI EA '16. San Jose, California, USA: ACM, 2016, 3558–3565. ISBN: 978-1-4503-4082-3. DOI: 10.1145/2851581.2856492 3.

[GLG*13] G., SAMUEL, L., ALEXANDER, G., NILS, et al. "LineUp: Visual Analysis of Multi-Attribute Rankings". *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)* 19.12 (2013), 2277–2286. DOI: 10.1109/TVCG.2013.173 3.

[H.99] H., MARK A. *Correlation-based Feature Selection for Machine Learning*. Tech. rep. 1999 4.

[Hol16] HOLZINGER, A. "Interactive machine learning for health informatics: when do we need the human-in-the-loop?": *Brain Informatics* 3.2 (2016), 119–131. ISSN: 2198-4026. DOI: 10.1007/s40708-016-0042-6 3.

[HS88] H., SANDRA G. and S., LOWELL E. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research". *Human Mental Workload*. Ed. by H., PETER A. and M., NAJMEDIN. Vol. 52. Advances in Psychology. North-Holland, 1988, 139 – 183. DOI: https://doi.org/10.1016/S0166-4115(08)62386-9 8.

[HY16] HE, Z. and YEN, G. G. "Visualization and Performance Metric in Many-Objective Optimization". *IEEE Transactions on Evolutionary Computation* 20.3 (2016), 386–402. ISSN: 1089-778X. DOI: 10.1109/TEVC.2015.2472283 3.

[HY17] HE, Z. and YEN, G. G. "Comparison of visualization approaches in many-objective optimization". *2017 IEEE Congress on Evolutionary Computation (CEC)*. 2017, 357–363. DOI: 10.1109/CEC.2017.7969334 3.

[JZF*09] JEONG, D. H., ZIEMKIEWICZ, C., FISHER, B., et al. "iPCA: An Interactive System for PCA-based Visual Analytics". *Computer Graphics Forum*. Vol. 28. 3. Wiley Online Library. 2009, 767–774 2.

[KBE14] KOMER, B., BERGSTRA, J., and ELIASMITH, C. "Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn". *ICML workshop on AutoML*. 2014 2, 3, 5.

[KBS*11] KULESZA, T., BURNETT, M., STUMPF, S., et al. "Where Are My Intelligent Assistant's Mistakes? A Systematic Testing Approach". *Proceedings of the Third International Conference on End-user Development*. IS-EUD'11. Torre Canne, Italy: Springer-Verlag, 2011, 171–186. ISBN: 978-3-642-21529-2 3.

[KCPE16] KIM, H., CHOO, J., PARK, H., and ENDERT, A. "InterAxis: Steering scatterplot axes via observation-level interaction". *IEEE transactions on visualization and computer graphics* 22.1 (2016), 131–140 2.

[KDJH08] KRESIMIR, M., DENIS, G., JELOVIC, M., and H., HELWIG. "Interactive Visual Steering – Rapid Visual Prototyping of a Common Rail Injection System". *IEEE Transactions on Visualization and Computer Graphics* 14.6 (Nov. 2008). URL: https://www.cg.tuwien.ac.at/research/publications/2008/Matkovic-2008-Steer/ 2.

[KEV*18] KWON, B. C., EYSENBACH, B., VERMA, J., et al. "Clustervision: Visual Supervision of Unsupervised Clustering". *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), 142–151. ISSN: 1077-2626. DOI: 10.1109/TVCG.2017.2745085 3.

[KKW*17] KWON, B. C., KIM, H., WALL, E., et al. "AxiSketcher: Interactive Nonlinear Axis Mapping of Visualizations through User Drawings." *IEEE Trans. Vis. Comput. Graph.* 23.1 (2017), 221–230 2.

[KLTH10] KAPOOR, A., LEE, B., TAN, D., and HORVITZ, E. "Interactive Optimization for Steering Machine Classification". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, 2010, 1343–1352. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753529 2, 3, 10.

[KPB14] KRAUSE, J., PERER, A., and BERTINI, E. "INFUSE: Interactive Feature Selection for Predictive Modeling of High Dimensional Data". *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), 1614–1623. ISSN: 1077-2626. DOI: 10.1109/TVCG.2014.2346482 4.

[KPHH11] KANDEL, S., PAEPCKE, A., HELLERSTEIN, J., and HEER, J. "Wrangler: Interactive Visual Specification of Data Transformation Scripts". *ACM Human Factors in Computing Systems (CHI)*. 2011 2.

[KTD17] KIM, S., TASSE, D., and DEY, A. K. "Making Machine-Learning Applications for Time-Series Sensor Data Graphical and Interactive". *ACM Trans. Interact. Intell. Syst.* 7.2 (July 2017), 8:1–8:30. ISSN: 2160-6455. DOI: 10.1145/2983924 3.

[KTH*16] KOTTHOFF, L., THORNTON, C., HOOS, H. H., et al. "Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA". *Journal of Machine Learning Research* 17 (2016), 1–5 3.

[LHM*13] LEMAN, S. C., HOUSE, L., MAITI, D., et al. "Visual to parametric interaction (v2pi)". *PLoS one* 8.3 (2013), e50474 2.

[LSL*17] LIU, M., SHI, J., LI, Z., et al. "Towards Better Analysis of Deep Convolutional Neural Networks". *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), 91–100. ISSN: 2160-9306. DOI: 10.1109/TVCG.2016.2598831 3.

[LWLZ17] LIU, S., WANG, X., LIU, M., and ZHU, J. "Towards better analysis of machine learning models: A visual analytics perspective". *Visual Informatics* 1.1 (2017), 48 –56. ISSN: 2468-502X. DOI: https://doi.org/10.1016/j.visinf.2017.01.006 4.

[LYL14] LI, M., YANG, S., and LIU, X. "Diversity Comparison of Pareto Front Approximations in Many-Objective Optimization". *IEEE Transactions on Cybernetics* 44.12 (2014), 2568–2584. ISSN: 2168-2267. DOI: 10.1109/TCYB.2014.2310651 3.

[MLa] ML, BIG. *Big ML*. https://bigml.com/. Accessed : March 20, 2019 3.

[MLb] ML, SCIKIT LEARN. *Scikit Learn Model Metrics API Reference*. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics. Accessed : March 20, 2019 5, 6.

[MLY17] MIQING, L., LIANGLI, Z., and Y., XIN. "How to Read Many-Objective Solution Sets in Parallel Coordinates". *CoRR* abs/1705.00368 (2017) 3.

[Mov] *Movies Dataset, howpublished = https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset#movie_metadata.csv, note = Accessed: 2018-12-11 8, 9*.

[NKHK17] NUSHI, B., KAMAR, E., HORVITZ, E., and KOSSMANN, D. "On Human Intellect and Machine Failures: Troubleshooting Integrative Machine Learning Systems". *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI'17. San Francisco, California, USA: AAAI Press, 2017, 1017–1025 3.

[Off] OFFICE, THE SAN FRANCISCO CONTROLLER'S. *Employee Compensation Data*. https://data.sfgov.org/City-Management-and-Ethics/Employee-Compensation/88g8-5mnd. Accessed : March 15, 2019 7–9.

[PBD*10] PATEL, K., BANCROFT, N., DRUCKER, S. M., et al. "Gestalt: Integrated Support for Implementation and Analysis in Machine Learning". *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*. UIST '10. New York, New York, USA: ACM, 2010, 37–46. ISBN: 978-1-4503-0271-5. DOI: 10.1145/1866029.1866038 2.

[PPJ07] PAPARIZOS, S., PATEL, J. M., and JAGADISH, HV. "SIGOPT: Using schema to optimize XML query processing". *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE. 2007, 1456–1460 3.

[PSTW*16] PAJER, S., STREIT, M., TORSNEY-WEIR, T., et al. "WeightLifter: Visual Weight Space Exploration for Multi-Criteria Decision Making". *IEEE Transactions on Visualization and Computer Graphics* (2016) 2.

[RAL*17] REN, D., AMERSHI, S., LEE, B., et al. "Squares: Supporting Interactive Performance Analysis for Multiclass Classifiers". *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), 61–70. ISSN: 1077-2626. DOI: 10.1109/TVCG.2016.2598828 3.

[RHY14] REN, D., HÖLLERER, T., and YUAN, X. "iVisDesigner: Expressive Interactive Design of Information Visualizations". *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), 2092–2101. ISSN: 1077-2626. DOI: 10.1109/TVCG.2014.2346291 2.

[RLD*18] RUDOVIC, O., LEE, J., DAI, M., et al. "Personalized machine learning for robot perception of affect and engagement in autism therapy". *Science Robotics* 3.19 (2018). DOI: 10.1126/scirobotics.aao6760 3.

[RSG16] RIBEIRO, M. T., SINGH, S., and GUESTRIN, C. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, 1135–1144. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939778 2–4.

[SAS02] SARAVANAN, R, ASOKAN, P, and SACHIDANANDAM, M. "A multi-objective genetic algorithm (GA) approach for optimization of surface grinding operations". *International Journal of Machine Tools and Manufacture* 42.12 (2002), 1327 –1334. ISSN: 0890-6955. DOI: https://doi.org/10.1016/S0890-6955(02)00074-3 10.

[SC11] SAHU, R. and CHATURVEDI, A. K. "Many-Objective Comparison of Twelve Grid Scheduling Heuristics". 2011 3.

[SHB*14] SEDLMAIR, M., HEINZL, C., BRUCKNER, S., et al. "Visual Parameter Space Analysis: A Conceptual Framework". *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), 2161–2170. ISSN: 1077-2626. DOI: 10.1109/TVCG.2014.2346321 2.

[SKBE17] SAKET, B., KIM, H., BROWN, E. T., and ENDERT, A. "Visualization by Demonstration: An Interaction Paradigm for Visual Data Exploration". *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), 331–340. ISSN: 1077-2626. DOI: 10.1109/TVCG.2016.2598839 2.

[SLT17] SUN, Y., LANK, E., and TERRY, M. "Label-and-Learn: Visualizing the Likelihood of Machine Learning Classifier's Success During Data Labeling". *Proceedings of the 22Nd International Conference on Intelligent User Interfaces*. IUI '17. Limassol, Cyprus: ACM, 2017, 523–534. ISBN: 978-1-4503-4348-0. DOI: 10.1145/3025171.3025208 3.

[SRL*07] STUMPF, S., RAJARAM, V., LI, L., et al. "Toward Harnessing User Feedback for Machine Learning". *Proceedings of the 12th International Conference on Intelligent User Interfaces*. IUI '07. Honolulu, Hawaii, USA: ACM, 2007, 82–91. ISBN: 1-59593-481-2. DOI: 10.1145/1216295.1216316 2.

[SRL*09] STUMPF, S., RAJARAM, V., LI, L., et al. "Interacting Meaningfully with Machine Learning Systems: Three Experiments". *Int. J. Hum.-Comput. Stud.* 67.8 (Aug. 2009), 639–662. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2009.03.004 3.

[SSZ*17] S., DOMINIK, S., MICHAEL, Z., LEISHI, et al. "What you see is what you can change: Human-centered machine learning by interactive visualization". *Neurocomputing* 268 (2017), 164–175 3.

[TCWM18] T., LI, CONVERTINO, G., WANG, W., and MOST, H. "HyperTuner: Visual Analytics for Hyperparameter Tuning by Professionals". *Machine Learning from User Interaction for Visualization and Analytics, IEEE VIS 2018* (2018) 3.

[TDCF07] TULLIO, J., DEY, A. K., CHALECKI, J., and FOGARTY, J. "How It Works: A Field Study of Non-technical Users Interacting with an Intelligent System". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. San Jose, California, USA: ACM, 2007, 31–40. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240630 3.

[THHLB13] THORNTON, C., HUTTER, F., HOOS, H. H., and LEYTON-BROWN, K. "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms". *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2013, 847–855 3.

[TLB*11] TAMUZ, O., LIU, C., BELONGIE, S., et al. "Adaptively Learning the Crowd Kernel". *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11. Bellevue, Washington, USA: Omnipress, 2011, 673–680. ISBN: 978-1-4503-0619-5 2–4.

[TLKT09] TALBOT, J., LEE, B., KAPOOR, A., and TAN, D. S. "EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers." *CHI*. Ed. by JR., DAN R. OLSEN, ARTHUR, RICHARD B., HINCKLEY, KEN, et al. ACM, 2009, 1283–1292. ISBN: 978-1-60558-246-7 2.

[VDEW11] VAN DEN ELZEN, S. and van WIJK, J. J. "Baobabview: Interactive construction and analysis of decision trees". *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*. IEEE. 2011, 151–160 2, 4.

[WEF13] WALKER, D. J., EVERSON, R., and FIELDSEND, J. E. "Visualizing Mutually Nondominating Solution Sets in Many-Objective Optimization". *Trans. Evol. Comp* 17.2 (Apr. 2013), 165–184. ISSN: 1089-778X. DOI: 10.1109/TEVC.2012.2225064 3.

[WFE12] WALKER, D., FIELDSEND, J., and EVERSON, R. "Visualising Many-objective Populations". *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*. GECCO '12. Philadelphia, Pennsylvania, USA: ACM, 2012, 451–458. ISBN: 978-1-4503-1178-6. DOI: 10.1145/2330784.2330853 3.

[WLDW01] WOLFMAN, S. A., LAU, T., DOMINGOS, P., and WELD, D. S. "Mixed Initiative Interfaces for Learning Tasks: SMARTedit Talks Back". *Proceedings of the 6th International Conference on Intelligent User Interfaces*. IUI '01. Santa Fe, New Mexico, USA: ACM, 2001, 167–174. ISBN: 1-58113-325-1. DOI: 10.1145/359784.360332 3.

[WN17] WENSKOVITCH, J. and NORTH, C. "Observation-Level Interaction with Clustering and Dimension Reduction Algorithms". *Proceedings of the 2Nd Workshop on Human-In-the-Loop Data Analytics*. HILDA'17. Chicago, IL, USA: ACM, 2017, 14:1–14:6. ISBN: 978-1-4503-5029-7. DOI: 10.1145/3077257.3077259 2.

[YL09] YEH, I. and LIEN, C. "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients". *Expert Systems with Applications* 36.2, Part 1 (2009), 2473 – 2480. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2007.12.020 8.

[YSCR18] YANG, Q., SUH, J., CHEN, N., and RAMOS, G. "Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models". ACM, 2018 2, 3.

[ZSZR18] Z., XIAOJIN, SINGLA, A., ZILLES, S., and RAFFERTY, A. N. "An Overview of Machine Teaching". *CoRR* abs/1801.05927 (2018) 2, 3.