Gaggle: Visual Analytics using Interactive Multi-Model Steering

Subhajit Das, Dylan Cashman, Remco Chang, Alex Endert

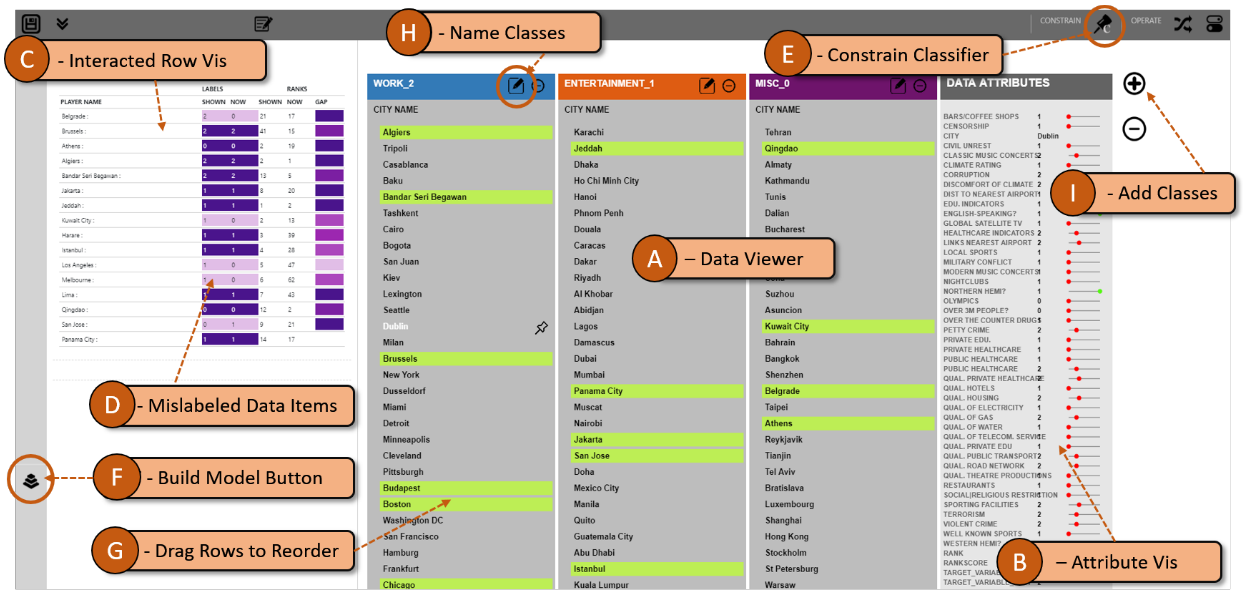


Fig. 1. The Gaggle user interface allowing people to steer multiple models for classification and ranking.

Abstract— Recent multi-model visual analytics systems make use of multiple machine learning models to better fit the data. This has been shown to be an improvement over traditional single-model approach in which systems rely on a single, pre-defined model that may lack the flexibility needed to capture nuances in the data. However, while multi-model visual analytic systems can be effective, their added complexity poses usability concerns as users are required to interact with the parameters of multiple models. In this paper, we present Gaggle, a multi-model visual analytics system that utilizes the advantages of steering multiple models while hiding the added complexity from the user. Specifically, user interaction with Gaggle is similar to existing single-model systems, in that users can demonstrate groups of similar data points for classification or partial ordering of data points for ranking models. Based on these user interactions, Gaggle generates multiple models but only presents the best model based on user preferences. As a result, this multi-model steering approach allows the user to benefit from the increased accuracy of multiple models while retaining the ease-of-use of a single-model system. To evaluate Gaggle, we conducted a controlled user study. The result confirms our hypothesis that Gaggle is easy to use and users can discover accurate models without being burdened with keeping track of multiple models. This finding suggests a possible new approach for visual analytics systems that we discuss in the paper.

Index Terms—visual analytics, user interaction, model steering, interactive machine learning

1 Introduction

Visual analytic techniques continue to leverage machine learning to provide people effective systems for gaining insights into data [23]. These systems help domain experts combine their knowledge and reasoning skills about a dataset or domain with the computational prowess of machine learning. For example, domain experts may understand nuances of the domain or datasets that are not emphasized by the model such as data quality issues, seasonal effects of temporal data, the increased importance of a small subset of the data otherwise washed out in the entirety of the whole dataset, or more. Thus, methods are needed that steer and parameterize models to incorporate user feedback into the machine learning algorithms. This paradigm of incrementally incorporating user input into models is often referred to as *model steering* [56].

Visual analytic techniques exist that allow users to steer models. For example, exposing model parameters through control panels give users direct control over aspects of models (e.g., iPCA [28], Clusterophile2 [15], Hawkeye [45]). More recently, an alternative method for model steering focuses on giving users the ability to visually demonstrate their intended changes to the model rather than controlling the model parameters directly (e.g., [13, 19, 22, 58, 59]). For instance, InterAxis [31] incorporates user feedback to adjust a linear dimension reduction model by letting people suggest similar data points that should be placed near each other. In both of these methods for user feedback, the mechanism for model steering adjusts the single model for which the system is designed and developed for. When this model is correctly chosen for the phenomena, task, data distribution, or question users try to answer, these existing techniques can effectively support users in exploration and analysis.

However, this "single-model" approach can be limiting in exploratory data analysis. For instance, what if the single model for

• *Subhajit Das and Alex Endert are with Georgia Institute of Technology. E-mail: das, endert@gatech.edu*
• *Dylan Cashman and Remco Chang are with Tufts University, USA E-mail: dcashm01@cs.tufts.edu, remco@cs.tufts.edu*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

which the tool is designed for is insufficient in modelling the specific dataset used? Alternatively, what if the exploratory task of the users changes over time, and thus a different model is better able to capture the phenomena being explored? Finally, how can visual analytic systems support more realistic analysis sessions that regularly require multiple tasks to be supported (e.g., classifying items into groups, then ranking the items within each class)?

In these situations, visual analytic techniques that utilize multiple models may be better suited. Such "multi-model" approaches allow users to traverse a greater set of possible models that might align better with their task or domain. For example, Clustervision [**?**] allows users to inspect multiple clustering algorithms and select one based on quality and preference. Similarly, Snowcat [14] allows inspecting multiple ML models across a diverse set of tasks such as classification, regression, time-series forecasting, etc.

However many of the current multi-model systems such as Snowcat do not support multi-model steering. This is likely because with the added computational advantage of using multiple models comes with added complexity in usability. For example, Clustervision [**?**] requires user input on model parameters, their properties, and metrics such as the Silhouette Coefficient, Calinski-Harabaz Index, Davies-Bouldin Index, etc. Thus, while existing multi-model systems are powerful, some do not support multi-model steering such as Snowcat and others can be difficult to use such as Clustervision, especially for users with limited understanding of machine learning or data science.

In this paper, we present Gaggle, a visual analytic system that uses simple user interactions to enable user feedback to a multi-model steering technique. Gaggle enables users to classify and rank data items, where each of these two tasks is supported by a multi-model steering approach. To keep the interactivity simple, the model steering interactions operate on data cases which serve as demonstrations to steer the models. For example, to steer classification models, users can drag data items into specific classes. Similarly, to steer ranking models, users can demonstrate that specific items should be higher or lower within a class. Gaggle incorporates user feedback using a multi-model steering technique. This multi-model steering technique applies user feedback to multiple models, by steering hyperparameters of multiple classification and ranking models within a single user interface. Given user feedback, Gaggle computes an optimal hyperparameter combination for both types of models using a Bayesian optimization strategy [39]. Further, our multi-model technique automatically selects the best model based on model metrics inferred from user interactions. This simplifies the complexity of steering multi-model systems, allowing users to interact with Gaggle as if it is a single-model system while retaining the benefits of using multiple models.

At first glance, it may seem that multi-model approaches are overwhelmingly advantageous over single-model alternatives. Since they can explore a broader space of model parameterizations and model types, it would seem that they should be able to generate a better fit and thus support exploration better. However, this claim has not been fully tested. Thus, our paper presents the results of a study that explores single-model vs. multi-model steering approaches. The purpose of the study is twofold. First, we want to understand the model switching behavior for multi-model steering. How often does the model change, and why? Second, does the multi-model steering technique produce better models? To minimize potential confounds caused by user interfaces, we used the Gaggle interface and interactions for both conditions, where the only difference between the two is the model steering approach.

The results of our study indicate that multi-model steering outperformed single-model steering for complex tasks such as classification and ranking of a multi-class dataset. However, both multi-model and single-model steering showed comparable performance for simple tasks such as binary classification and rankings tasks. Furthermore, the study revealed when during the analysis process this switching occurred.

Overall, the contributions of this paper include:

- A visual analytic system (Gaggle) which provides either single-model or multi-model steering for classification and ranking using simple, demonstration-based user interactions.
- A multi-model steering technique facilitating Bayesian optimization based hyperparameter tuning and automated model selection.
- The results of a user study comparing single-model and multi-model steering for classification and ranking tasks.

2 Related Works

2.1 Interactions in Visual Analytics

Interactive model construction has been a flourishing avenue of research in the recent past. In general, the design of such systems make use of both explicit user interactions such as specifying parameters via graphical widgets (e.g., sliders), or implicit feedback including demonstration-based interactions or eye movements to provide guidance on model selection and steering. These types of systems build many kinds of models, including metric learning [13], decision trees [55], and dimensional reduction [22, 31, 34]. For example, Jeong et al. presented iPCA to show how directly manipulating the weights of attributes via control panels helps people adjust principal component analysis [28]. Similarly Amershi et al. presented an overview of interactive model building [4]. Our work differs from these models in two primary ways. First, our technique searches through multiple types of models (i.e., Random Forest models with various hyperparameter and parameter settings, as well as ranking models). Second, our tool interprets user interaction as feedback on all models causing hyperparameter tuning directly changing model behavior *in parallel*.

Stumpf et al. conducted experiments to understand the interaction between users and machine learning based systems [50]. Their results showed that a collaborative shared intelligence-based framework grounded in user interactions can help both users and systems. Stumpf et al. conducted a think-aloud study to understand the forms of feedback humans might give to machines [49]. They found that these included suggestions for re-weighting of features, proposals for new features and feature combinations, relational features, and wholesale changes to the learning algorithm. They showed that user feedback has the potential to improve ML systems, but that learning algorithms need to be extended to assimilate this feedback [49].

Interactive model steering can also be done via demonstration-based interaction. The core principle in these approaches is that users do not adjust the values of model parameters directly, but instead visually demonstrate partial results from which the models learn the parameters [11, 13, 20–22, 25, 35]. For instance, Brown et al. showed how repositioning points in a scatterplot can be used to demonstrate an appropriate distance function [13]. It saves the user the hassle to manipulating model hyperparameters directly to reach their goal. Similarly, Kim et al. presented InterAxis [31], which showed how users can drag data objects to the high and low locations on both axes of a scatterplot to help them interpret, define, and change axes with respect to a linear dimension reduction technique. Using this simple interaction, the user can define constraints which informed the underlying model to understand how the user is clustering the data. Wenskovitch and North used the concept of observation level interaction in their work by having the user define clusters in the visualized dataset [59]. By visually interacting with data points, users are able to construct a projection and a clustering algorithm that incorporated their preferences. Finally, prior work has shown benefits from directly manipulating visual glyphs to interact with visualizations, as opposed to control panels [10, 21, 30, 36, 42, 44]. From a user experience perspective, the work presented in this paper aligns closely with these demonstration-based techniques. Gaggle's interaction design does not presume expertise in model building or steering, but rather lets users manipulate the visual results of the models to incrementally refine and steer them.

2.2 Multi-Model Visual Analytic Systems

Das et al. showed interactive multi-model inspection and steering of multiple regression models [17]. Hypertuner [54] looked at tuning multiple machine learning model's hyperparameters. Sedlmair et al. [47] defined a method of variation of model parameters, generating a diverse range of model outputs for each such combination of parameters. This technique called visual parameter analysis investigated the relationship between the input and the output within the described parameter space. Similarly Pajer et al. [37] showed a visualization technique enabling visual exploration of the weight space which ranks plausible solutions in the domain of multi-critieria decision making. Kwon et al. [**?**] demonstrated a technique to visually identify and select an appropriate cluster model from multiple clustering algorithms and parameter combinations. However they target data scientists, while Gaggle is designed for users who are non-experts in ML.

Clusterophile 2 [15] enabled users to explore different choices of clustering parameters and reason about clustering instances in relation to data dimensions. Another system StarSpire from Bradel et al. [11] showed how semantic interactions [21] can steer multiple text analytic models. While effective, their system is scoped to text analytics and handling text corpora at multiple levels of scale. In contrast, our work focuses on tabular data, and steering of multiple models within two classes of models (classification and ranking). Further, our work also steers hyperparameters of each of these models.

2.3 Human-Centered Machine Learning

Human-Centered Machine Learning focuses on how to include people in ML processes [4–6, 43]. A related area of study is the modification of algorithms to account for human intent. Sacha et al. showed how visual analytic based processes can allow interaction between automated algorithms and visualizations for effective data analysis [43]. They examined criteria for model evaluation on an interactive supervised learning system. The found users evaluate models by conventional metrics, such as accuracy and cost, as well as novel criteria such as unexpectedness. Sun et al. developed Label-and-Learn, allowing users to label data facilitated by interactive visualizations [51]. Their goal was to allow users to determine a classifier's success, and to analyze the performance benefits of adding expert labels [51]. Bernard et al. emphasized the knowledge generation process of users performing visual interactive labeling tasks, as opposed to conventional machine learning methods [8, 9]. Ren et al. explained debugging multiple classifiers using an interactive tool called Squares [41].

Holzinger et al. discussed how automatic machine learning methods are useful in numerous domains [26]. They note that these systems generally benefit from large static training sets, which ignore frequent use cases where extensive data generation would be prohibitively expensive or unfeasible. In the cases of smaller datasets or rare events, automatic machine learning suffers from insufficient training samples. They claim, such an NP-hard problem can be successfully solved by interactive machine learning via input and assistance from a human agent [26, 27]. This concept of computational models fostering human and machine collaboration are further explored in [16]. In our system and technique description, we extend these formalizations by considering human interaction as an estimation of the loss function of the models viewed by the user. In doing so, we generalize human centered machine learning to multiple models.

2.4 Automated Model Selection

Model building requires selecting a model type, finding a suitable library, and then searching through the hyperparameter spaces for an optimal setting to fit their data. For non-experts, this task can amount to many iterations of trial and error. In order to combat this guessing game, non-experts could use automated model selection tools such as AutoWeka [33, 53], SigOpt [38], HyperOpt [7, 32], and AUTO-SKLEARN [24]. These tools execute intelligent searches over the model space and hyperparameter spaces, providing an optimal model for the given problem type and dataset. However, these tools are all based on optimization of an objective function which takes into account only features or attributes that are quantifiable, often ignoring user feedback. Instead, our work explores how to incorporate domain expertise into the model selection process through user feedback.

3 Motivation

Multi-model steering techniques facilitate the adjustment of model hyperparameters to incrementally construct models that are better suited to user goals. In this paper, we consider the common problem of datasets that either lack adequate ground truth, or do not have it [40, 52, 60]. To resolve this problem, Gaggle allows users to iteratively define classes and add labels. On each iteration, users add labels to data items and then build a classification model. Further, users are able to rank data items within each class based on criteria relevant to their task or domain.

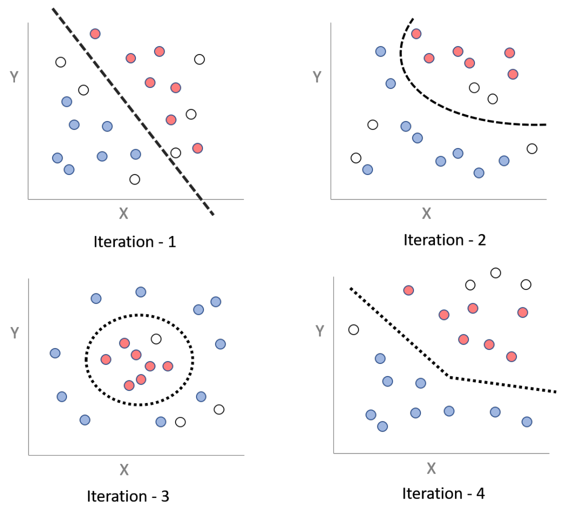


Fig. 2. A hypothetical binary classification problem shows how different model hyperparameters may be needed to model the changing user interest at each iteration. Blue and orange points represent positive and negative classes; white points represent data items not interacted with.

The need for multi-model steering is exemplified in this iterative exploration and model building scenario. During this process, users may change their task definition slightly, or learn new information about their data. In these cases, their user feedback may be better modelled by a different model hyperparameterization than their feedback earlier in the process. Updating the class definition or showing better examples directly affects the underlying decision boundary which the classifier needs to map correctly. For example, in the first iteration, a linear decision boundary might characterize the data. However, when new examples for classes are provided the decision boundary might be mapped only using a polynomial or radial surface (Refer Figure 2). In situations like this, a multi-model steering approach benefits the user by adjusting model hyperparameters and then automatically selecting a model which is most appropriate for the task, the label distribution, and the underlying decision boundary. In contrast, a single-model steering technique will struggle based on the pre-selected hyperparameters. For example, an SVM with a linear kernel would inadequately characterize a non-linear decision boundary.

4 Model Search Space and Steering

This section describes core concepts and terminology used.

4.1 Models

We define a *model* as a function $f : \mathscr{X} \mapsto \mathscr{Y}$, mapping from the input space $\mathscr{X}$ to the prediction space $\mathscr{Y}$. We are concerned primarily with *semi-supervised learning* models, in which we are provided with a partially labeled or un labeled training set $D_{train} = D_U \cup D_L$, where D_L is labeled data and D_U is unlabeled data such that if $d_i \in D_L$, then $d_i = (\mathbf{x_i}, y_i)$, and if $d_i \in D_U$, then $d_i = (\mathbf{x_i})$, where $\mathbf{x_i}$ are features and y_i is a label.

A *learning algorithm* A maps a training set D_{train} to a model f by searching through a parameter space. A model is described by its *parameters* θ, while a learning algorithm is described by its *hyperparameters* λ. A model parameter is internal to a model, where its value can be estimated from the data. Model parameters refer to values that are learned during training of a model, such as coefficients, while hyperparameters are typically determined through some process external to training, such as cross validation.

Automatic model selection algorithms search across the set of hyperparameter spaces Λ $\lambda \in \Lambda$, that correspond to a particular learning algorithm, and return the model $f^* = A_\lambda(D_{train})$ that minimizes a loss

Table 1. User tasks, learning algorithms, hyperparameters, and parameters in Gaggle.

Tasks	Learning Algorithm	Hyperparameters	Parameters
Classification	Random Forest	Criteria Max Depth Min Samples	Attribute Entropy, Information Gain
Ranking	Ranking Random Forest	Criteria Max Depth Min Samples	Attribute Entropy, Information Gain

function $\mathscr{L}(f)$ such as accuracy or mean squared error. In contrast, visual analytics systems such as Interaxis [31] and Dis-Function [13] search over the parameter space Θ of a particular learning algorithm that already has its hyperparameters defined. These systems incorporate user input to steer model selection by choosing a loss function $\mathscr{L}(f,i)$ that is dependent on a human interaction i. Our system Gaggle incorporates user input to search over multiple learning algorithms. The interactions i that affect our loss function are labelings and rankings of data points, and learning algorithms are the set of classification and ranking algorithms.

4.2 Model Search Space

Here we describe *model search space*, which we define as the complex possible combinations of different learning algorithms and hyperparameter values. For example, SVM, Naive Bayes, and Decision Trees are some of the learning algorithms but there are many more. Every algorithm has a known set of hyperparameters, which can take values within a domain range. The space of learning algorithms and hyperparameters form a potentially infinite *model search space*. Our definition of *model search space* is related to the work by Brown et al. [12] where they presented a tool called ModelSpace to analyze how the model parameters have changed over time during data exploration.

Gaggle uses Random Forest and thus builds multiple Random Forest models by taking hyperparameter combinations within a set domain range. Gaggle samples across three hyperparameters of Random Forest (criteria, max depth, min samples to set a node as a leaf) to generate new models. Specifically, we sample the hyperparameter "criteria", which can have the value 'gini', or 'entropy', the "max depth" parameter, D, which represents the maximum depth of the tree, min sample leaf (see Table 1). While Gaggle uses a Random Forest model for the quantitative and qualitative system evaluation (explained later in the paper), the general optimization method used is designed to work with other learning algorithm and hyperparameter combinations as well. Gaggle's optimizer can build multiple SVM models using a set of chosen hyperparameters such as kernel type, C-value, etc.

4.3 Model Ranking and Selection

Many models can be created by traversing the model search space. Exposing all of these models to the user might be overwhelming. The system ranks $n = 200$ models in the search space by a loss function $\mathscr{L}(A_{\lambda}, D_{train}, D_{valid})$ which returns a single model based on metrics such as training accuracy, correctly labeled interacted data items, and correct ranked ordering of interacted data items (Figure 5). This optimal model is applied to the dataset and shown to the user.

We call our approach semi-automatic because even though the system automatically selects the optimal model, they are evaluated based on metrics the user specifies by interacting and exploring the data. This approach is well-suited for situations where there is no ground truth, the quality of training data is questionable, or the end user knows more about the data than what is explicitly contained in the data. In those cases, semi-automatic human-in-the-loop based strategies have the potential to build models which accurately represent the user's expertise and knowledge. In Gaggle, we infer the value of the loss function based on the user interactions with the data and model predictions.

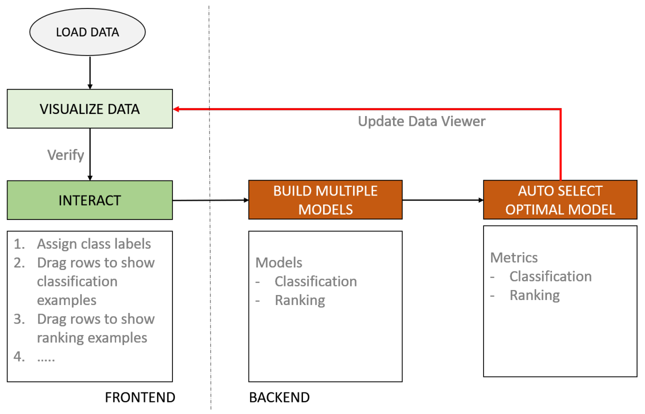


Fig. 3. Diagram highlighting Gaggle's workflow, also exemplified in the described usage scenario.

5 Usage Scenario

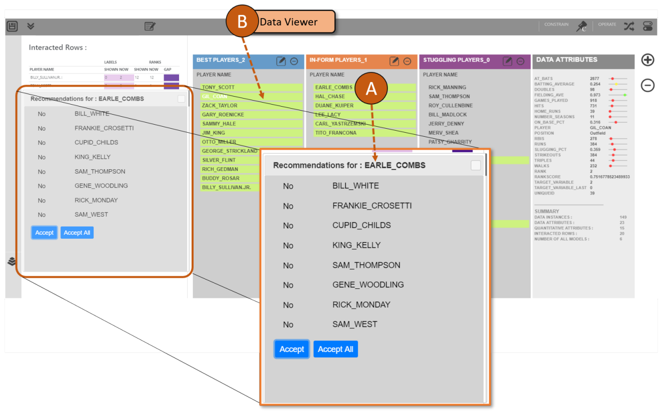
The tasks of classification and ranking are frequent problems faced by domain experts while exploring and making sense of their data. In support of this task, Gaggle allows users to assign data points to classes and then partially order data items within the classes to demonstrate classification and ranking. Next, the system responds by building multiple variants of classification and ranking models. Gaggle optimizes the machine learning models to automatically find an optimal model from the model search space based on performance criteria (explained in the sections below). This process continues until the users are satisfied with the model, meaning that the chosen model has correctly learned the user's subjective knowledge and interpretation of the data (Figure 3). At every iteration, users provide feedback to the system through various forms of interaction (e.g., dragging rows, assigning new examples to the class labels, correcting previous labels, etc.).

Problem Space: Imagine Jonathan runs a sports camp for baseball players. He has years of experience in assessing the potential of players. He not only understands which features from the data are important to his assessments but also has prior subjective knowledge about the players. In his day to day work, Jonathan needs to judge areas in which the players need further improvement. He would like to do this by placing players into different categories: "Best Players", "In-form Players" and "Struggling Players".

User-Provided Labeling: Jonathan starts by importing the dataset of baseball players (data publicly available from OpenML [57]). The data contains 400 players (represented as rows) and 17 attributes of both categorical and quantitative types. The dataset does not have any ground truth labels. He sees the list of all the players in the Data Viewer (Figure 4-B). He creates the three classes mentioned above and drags respective players in these bins or classes to add labels. Knowing Ernie Banks and, Carl Yastrzemski as very highly rated players, he places them in the "Best Players" class. Gaggle shows him recommendations of similar players in order to facilitate faster labeling. (Figure 4).

Automated Model Generation: Jonathan clicks the build model button from the Side Bar (Figure 1-F). Based on Jonathan's interaction so far, Gaggle runs its optimizer and automatically finds the best performing model, out of an exhaustive search of over 150 models. When the system responds, Jonathan continues his analysis of the data. He finds player Ernie Banks is mis-classified and places him in the "In-form Players" class instead of the "Best Players". He moves Ernie Banks and similar other mis-classified players to the correct class label and asks Gaggle to find an optimal model that takes his feedback into account.

Gaggle generates a new model and shows Jonathan the updated state of the training data in the data viewer. He reviews the results to find that many of the previously mis-classified players are correctly labeled, and pins them to ensure they do not change labels in future iterations Next, he looks at the attribute viewer (Figure 1-B) in search of players with high "batting average" and "home runs" values. He moves players that match his criteria into respective labels (e.g., placing "Sam West" and "Bill Madock", and in the "In-Form Players" class). After the



Recommendations for : EARLE_COMBS

No	BILL_WHITE
No	FRANKIE_CROSETTI
No	CUPID_CHILDS
No	KING_KELLY
No	SAM_THOMPSON
No	GENE_WOODLING
No	RICK_MONDAY
No	SAM_WEST

Fig. 4. Gaggle's recommendation dialog box.

model recomputes, he verifies the results returned by the model in the interacted row visualization (Figure 1-C). He accepts the classification model and moves on to rank the players within each class.

For each class, Jonathan drags players up and down to demonstrate his understanding of the ranking of players within classes. He iterates to check the updated optimal ranking model built by Gaggle. He checks the interacted row visualization to find where the model ranked each of the players he interacted with. It shows him expected player rank and assigned player rank (by the current model). He moves player "Norm Cash" and "Walker Cooper" to the top of the "struggling players" class, and moves player "Hal Chase" in the "best players" class down. He iterates further and sees that most of the players are relatively at the correct ranked spot. As a result, Gaggle helped Jonathan to classify and rank players solely based on his prior subjective domain knowledge, following the iterative process shown in Figure 3.

6 Gaggle: System Description

6.1 User Interface

Data Viewer. The main view of Gaggle is the Data Viewer which shows the data items within each class (Figure 1-A). Users can add, remove, or rename classes at any point during data exploration and drag data instances to bins to assign labels. Users can also re-order instances by dragging them higher or lower within a bin to specify relative ranking order of items. Gaggle marks these instance with a green highlight, see Figure 1-G. When Gaggle builds models and finds an optimal model, the Data Viewer updates the class membership and ranking of items. Our design decision to solve for a single model to show at each iteration is to simplify the user interface by removing a model comparison and selection step.

Attribute Viewer. Users can hover over data items to see attribute details (Figure 1-B) on the right. Every quantitative attribute is shown as a glyph on a horizontal line. The position of the glyph on the horizontal line shows the value of the attribute in comparison to all the other data instances. The color encodes the instance's attribute quality in comparison to all other instances (i.e., green, yellow, and red encodes high, mid, and low values respectively).

Data Recommendations. When users drag data instances to different bins, Gaggle recommends similar data instances which can also be added (Figure 4). This is to expedite class assignment during the data exploration process. The similarity is computed based on the total distance D_a of each attribute d_i of the moved data instance to other instances in the data. Users can accept or ignore these recommendations.

Interacted Row Visualization. This view (Figure 1-C) shows the list of all interacted data items. In addition, with color encoding it shows correct label matches (shown in blue color) and incorrect label matches(shown in pink color). Same is true for ranking (blue for correct ranked order prediction as expected and pink for otherwise. It helps users to know how many constraints were correctly predicted.

User Interactions. Gaggle lets users give feedback to the system to sample models in the next iteration, adjust model parameters and

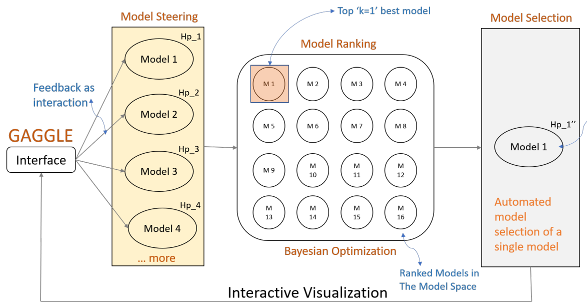


Fig. 5. The multi-model steering technique used in Gaggle.

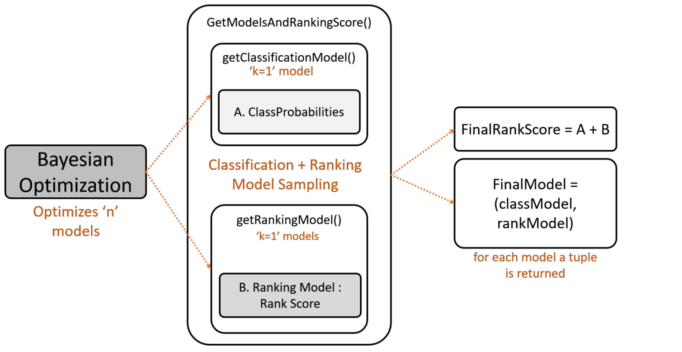

GetModelsAndRankingScore()
getClassificationModel()
'k=1' model
A. ClassProbabilities
Classification + Ranking Model Sampling
getRankingModel()
'k=1' models
B. Ranking Model : Rank Score
Bayesian Optimization
Optimizes 'n' models
FinalRankScore = A + B
FinalModel = (classModel, rankModel)
for each model a tuple is returned

Fig. 6. The model sampling and optimization approach for classification and ranking used in Gaggle.

hyperparameters, and allow users to explore data and gain insight.

- **Assign Class Labels**: Users can reassign classes by dragging data items from one class to another. They can also add or remove classes. These interactions provide more constraints to steer the hyperparameters of the classification model.
- **Reorder Items within Classes**: Users can reorder data items within classes as shown in Figure 1-G to change their ranking. This interaction helps users exemplify their subjective order of data instances within classes. This feedback is incorporated into the models as training data for the Ranking model.
- **Pin Data Items**: When sure of a class assignment of a data item, the user can pin it to the respective class bin. It ensures that data item will always be assigned that class in every subsequent iterations.
- **Constrain Classification Model**: When satisfied by the classification model, users can constrain the last best classifier. It allows users to move on to show ranking examples for Gaggle to focus on improving the ranking model (See Figure 1-C).

6.2 Multi-Model Steering Technique

This section describes the computational techniques which enable Gaggle to steer multiple models based on user feedback. Gaggle uses a multi-model steering approach for both classification and ranking tasks. When users classify data items, Gaggle constructs multiple classifiers and finds the best one using Bayesian optimization. When ranking, Gaggle constructs multiple ranking models and then finds the best one based on ranking metrics described in this section.

Bayesian Optimization Technique: To facilitate the interactive user feedback and steering of hyperparameters, Gaggle uses a Bayesian optimization technique [39, 48] to search and sample models (for both classification and ranking tasks) from the model search space as shown in Figure 5. Gaggle seeds the optimization technique by providing: a learning algorithm A, a domain range D_r for each hyperparameter, the total number of models to sample n, and how many models to visualize k. The Bayesian optimization module randomly picks a hyperparameter combination hp_1, hp_2 and hp_3. For example, a model

M_1 can be sampled by providing "criteria type" = *gini*, "max-depth" = 30, and "min-samples-leaf" = 12 ("criteria type", "max-depth", and "min-samples-leaf" are the hyperparameters used in Gaggle). Likewise, Bayesian optimization samples M_1, M_2, M_3, M_4 ... M_n models. For each such model, it also computes and stores the cross-validation score defined as cv_1,cv_2,cv_3, ... cv_n.

Bayesian optimization uses a Gaussian process to find an expected improvement point in the search space over current observations. For example, a current observation could be mapped to a machine learning model, and its metric for evaluation of the expected probability will be the model's cross-validation score. Using this technique, the optimization process ensures new sampled models improve over the currently best-known observation or model. Next, the Bayesian optimization module finds the model with the best score and updates the remaining views of the system (see Figure 5). Gaggle performs this process for both classification and ranking models.

Classification Model Technique: Gaggle begins with an unlabeled dataset. As the user interacts with a dataset of n items, labels are added. For example, if the user interacts with f data items, they become part of the training set for the classification model. The rest of the instances $n-f$, are used as a test set to assign labels from the trained model. If f is lower than a threshold value t, then Gaggle automatically finds s similar data instances to the interacted items and places them in the training set along with the interacted data items. The similarity is measured by the cosine distance function using the features of the interacted samples. This ensures there is enough training samples to train the classification model effectively. As the user iterates and interacts with more data instances, the size of the training set grows and test set shrinks, helping build a more robust classifier. For each classification model, Gaggle also determines the class probabilities P_{ij}, representing the probability of item i classified into class j.(e.g., $P_{10}, P_{20}, P_{11}, P_{21}, P_{31}, P_{41}, ...etc.$) The class probability is used to augment the ranking computation as they represent the confidence the model has over a data instance to be member of a said class.

Ranking Model Technique: Gaggle's ranking model is inspired by [29,58] which helps users to subjectively rank multi-attribute data instances. However, unlike these works, we used a Random Forest model (a similar approach to [61]) to classify between pairs of data instances R_i and R_j. The model predicts if R_i should be placed above or below R_j. We do the same between all the interacted data samples and the rest of the data set. Besides, we augmented it with a feature selection technique based on the interacted rows. For example assume a user moves R_i from rank B_i to B_j where $i > j$ (the row is meant to have a higher rank). The feature selection technique checks all the quantitative attributes of R_i and retrieves $m = 3$ (the value of m is learnt by heuristics) quantitative attributes Q_1, Q_2 and Q_3 which best represents why R_i should be higher in rank than R_j These features are the ones in which R_i is better than R_j. If $i < j$ (or if the row was meant to have a lower rank) Gaggle again retrieves $m = 3$ features. These features are the ones which best represents why R_i should be lower in rank than R_j. We do the same for all the interacted rows and finally we get a set of features (F_s, by taking the common features from each individually interacted row) which defines the user's intended ranked order. In this technique, if a feature satisfies one interaction but fails on another, they are left out. Only the common features across interacted items gets selected. The set of selected features F_s was then used to build the Random Forest model for Ranking. Using the classification models (class probabilities) and the ranking models Gaggle ranks the data instances within each class. A ranking model assigns a ranking score E_{ij} to each data instance by which they are sorted (ith instance, of jth class). A final ranking score is computed by combining the ranking score of a data instance R_i and its class probability P_{ij}, derived from the classification model. It is represented as $R_{ni} = E_{ij} * W_r + P_{ij} * (1 - W_r)$ where R_{ni} is new rank, W_r is the weight of the rank score and $1 - W_r$ is the weight of the classification probability (See Figure 6). The weights represent the proportion by which the class probabilities affect the ranking score, and sum to 1.

6.3 Model Ranking and Model Selection

This section describes how our multi-model steering technique ranks multiple ML models and automatically selects the best model. Gaggle selects an optimal model from a pool of models based on the following metrics which describe each model's performance:

Classification Metrics: Metrics used to evaluate the classification models include: percentage of wrongly labeled interacted data instances C_l, cross-validation scores from 10-fold evaluation C_v, and accuracy of the whole dataset C_a. The final metric is the sum total of these components computed as, $C_l * W_l + C_v * W_v + C_a * W_a$ where W_l, W_v, W_a are the respective weights for each of the aforementioned classification metric components. Different weight values were tested for both single and multi-model steering technique. Finally we chose the set of weights which led to the best gain in model accuracy.

Ranking Metrics: To evaluate the ranking models, Gaggle computes three ranking metrics based on the absolute distance from the instance's position before and after a said model M_i is applied to the data. Assume a row r is ranked q when the user interacted with the data. After applying model M_i to the data, the row r is at position p, then the absolute distance is given by $D_r = abs(p - q)$. The first ranking metric computes the absolute distances only between the interacted rows. It is defined as $D_u = \sum_{r \in R_i} d_r$ where row r is in the set R_i of all interacted rows. The second metric, D_v, computes the absolute distance between the interacted rows and the immediate h rows above and below of each interacted rows. In Gaggle, h defaults to 3 (but could be adjusted). The third metric, D_w, computes the absolute distance between all the instances of the data before and after a model is applied. defined as $D_w = \sum_{r \in R} d_r$ where row r is in the set R of all rows. A lower distance represents a better model fit. The final ranking metric is computed by the weighted summation of these metrics defined as $D_{total} = D_u * W_u + D_v * W_v + D_w * W_w$, where, W_u, W_v, W_w are the weights for the three ranking metrics. Weight values were tested for both single and multi-model steering techniques, and chosen based on the set of weights which gave the best model accuracy.

7 User Study

The goal of the study is to compare the multi-model steering (MMS) technique presented in this paper with a single-model steering (SMS) technique for a classification and ranking task. We tested the two model steering techniques using Gaggle. Both conditions use Gaggle's user interface and interactions to minimize the potential confounds caused by used different tools, and isolate the effect of MMS.

For the SMS condition, Gaggle used a single model steering technique with a pre-selected ML model. The hyperparameters for this model were chosen by training the model with a preset target label as the ground truth, using SK-Learn's random search technique [46] to find a hyperparameter combination with the best cross-validation score. This provided the SMS condition with a realistic starting model for the task. The MMS technique performed multi-model steering, as described in Section 6.2.

The primary research questions our study seeks to answer are:

Q1 How and when does model switching occur in the MMS condition?

Q2 How does performance compare between the MMS and SMS conditions?

Overall, we hypthesize that MMS will outperform SMS in our study, as it searches through a larger model space in each iteration, and should thus fit the user's preferences more closely. To understand this in more detail, our study tests the following hypotheses:

H1 MMS will outperform SMS (with respect to accuracy) for the combined task of classification (both multi-class and binary) and ranking.

H2 For multi-class classification and ranking problems, MMS will outperform SMS with respect to accuracy.

H3 For binary classification and ranking problems, MMS will outperform SMS with respect to accuracy.

7.1 Participants

We recruited 22 graduate and undergraduate students (14 male, 8 female). The only inclusion criteria were that participants should be non-experts in ML, and have adequate knowledge of movies and cities (datasets used for the study). All participants rated themselves fluent in English. None of the participants used Gaggle prior to the study. We compensated the participants with a $10 Amazon gift card. The study was conducted in a quiet lab environment using a laptop with a 17-inch display and a mouse. The full experiment lasted 60-70 minutes.

7.2 Study Design

The experiment with Gaggle was a 2 $\times$ 2 within-subjects study. Participants used Gaggle with SMS and MMS techniques to classify and rank data items. For each technique, participants were asked to complete 4 tasks: multi-class classification of items (3 classes), ranking the classified data items, binary classification of items, and ranking the classified data items. To reduce learning and ordering effects, both of the model steering techniques (SMS and MMS) and the dataset choice (Movies [3] and Cities [2]) were randomized for each trial. In total, each participant performed 8 tasks, 4 per model steering technique.

7.3 Tasks and Procedure

We began each study with a practice session to teach users about the workflow and interaction capabilities of the system. During this session, participants performed 4 tasks (multi-class classification + ranking and binary classification + ranking) on a Cars dataset [1]. We proceeded to the experimental sessions only when participants were confident enough to correctly use the system. For each trial, we randomized the order of SMS and MMS to minimize ordering effects. For each technique (SMS and MMS) participants built a multi-class classifier first. This was followed by a binary classification and ranking task on the same dataset. Then they repeat the same set of tasks on the alternate dataset. The study used two datasets, Movies [3] and Cities [2]. The movies data had 210 items, with 11 attributes while the cities dataset had 140 items with 45 attributes. We asked participants to create specific classes for each dataset. For the Movies dataset multi-class labels were *sci-fi*, *horror/thriller*, and *misc*, while for the Cities dataset multi-class labels were *fun-cities*, *work-cities*, and *misc*. For the binary classification task, the given labels were *popular* and *unpopular* (for Movies dataset), and *western* and *non-western* (for Cities dataset).

After each set of tasks, we asked users to answer a questionnaire.. We asked users to tell us in which class a sample of 15 data points (either Cities or Movies, depending on trial) should appear to test how well the user's categorization matches that of the model. Then, we ask them to provide the relative rank order of these data items, and compared that to the model's ranking. After completing both sessions, we conducted a semi-structured interview, where we asked open-ended questions, including: What was your experience using Gaggle? Tell me what intrigued you about the interactions? Which model do you think performed better (SMS or MMS)? Which interactions were helpful or confusing to perform the designated tasks?

7.4 Data Collection and Analysis

For quantitative assessment, we primarily rely on log data which stores model hyperparameters per iteration, class labels of the data items per iteration, the model's learning algorithm, data items users interacted with, etc. We also collected screen and audio recordings. We also collected qualitative feedback through a semi-structured interview.

We considered five dependent variables for this study: *Model accuracy (classification):* the accuracy of the model in predicting correct labels, *Model accuracy (ranking):* the accuracy of the ranking model *Perceived accuracy (classification):* number of correctly labeled data items (as obtained from the 15 data points we ask users to label), *Perceived accuracy (ranking):* number of correctly ranked data items (as obtained from the questionnaire after each session), and *User preference:* preference of SMS or MMS for various tasks (see Figure 9) provided by the user as a feedback from the questionnaire.

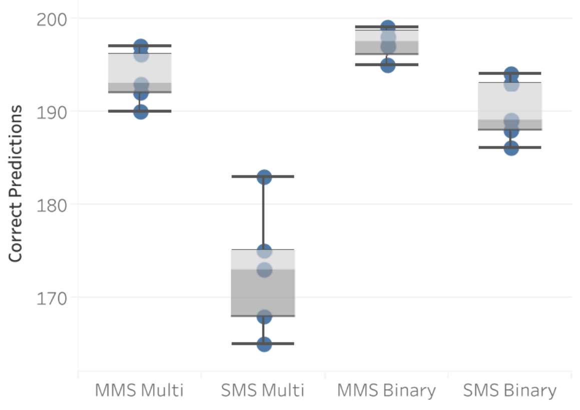


Fig. 7. The number of correctly predicted labels for interacted data items.

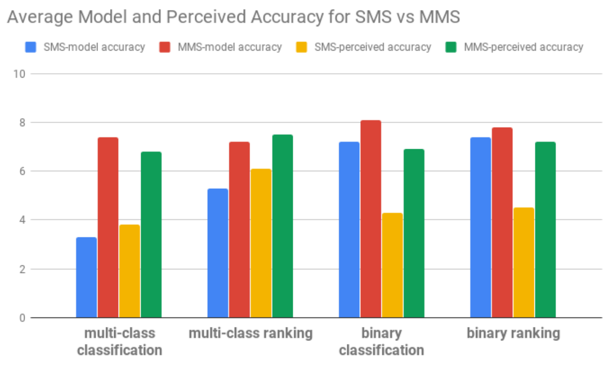


Fig. 8. Average Model and Percieved accuracy of SMS and MMS.

7.5 Quantitative Study Results

Defining Task Accuracy. We compare the two model steering techniques (SMS and MMS) with respect to which one is more capable of building a model which satisfies the user constraints. We analyze and report accuracy for each task (ranking and classification) using the metrics *model accuracy* and *perceived accuracy*. We compared the model accuracy and perceived accuracy for each interface and tested for statistically significant differences. Figure 8 shows model and perceived accuracy for each task.

Task Accuracy Across All Tasks. To test **H1**, we conducted a Friedman Test for Repeated-Measures and found a significant difference in model accuracy between the interface type SMS ($M = 0.512$ $[0.477, 0.547]$) and MMS ($M = 0.786$ $[0.732, 0.840]$) for all four tasks combined. Post-hoc Wilcoxon signed-rank tests with Bonferroni correction found statistical significance (M($p < 0.05$)). We used the Friedman Test for Repeated-Measures as it is a good indicator of statistical significance for multi-class classifiers with multiple datasets as suggested by [18]. Similarly, we conducted Friedman Test for Repeated-Measures with Post-hoc Wilcoxon signed-rank tests for perceived accuracy for SMS ($M = 0.410$ $[0.406, 0.414]$) and MMS ($M = 0.721$ $[0.708, 0.734]$). We found MMS significantly outperformed SMS for all tasks (M($p < 0.05$)), confirming **H1** (see Table 3).

Task Accuracy for Multi-class Classification. To test **H2**, we conducted a Friedman Test for Repeated-Measures using model accuracy between the two conditions to determine effects specifically on multi-class classification and ranking tasks. The results show that participants performed significantly better with MMS ($M = 0.824$ $[0.821, 0.827]$) than the SMS ($M = 0.623$ $[0.618, 0.628]$) for multi-class classification with M($p < 0.05$). Similarly, results for the multi-class ranking indicate M($p < 0.05$). Then we conduct a Friedman Test for Repeated-Measures between the two interfaces to determine effects on multi-class classification and ranking tasks with respect to perceived accuracy. These

Table 2. The change in hyperparameters in MMS and change in cross validation score per iteration for both SMS and MMS

Iter.	SMS Score	SMS Hyperparam	MMS Score	MMS Hyperparam

1	0.76	MaxDepth = 4
		Criteria = 'entropy'
		MinSamples = 10
			0.58	MaxDepth = 2
				Criteria = 'entropy'
				MinSamples = 3

2	0.45	MaxDepth = 4
		Criteria = 'entropy'
		MinSamples = 10
			0.55	MaxDepth = 12
				Criteria = 'gini'
				MinSamples = 8

3	0.45	MaxDepth = 4
		Criteria = 'entropy'
		MinSamples = 10
			0.62	MaxDepth = 22
				Criteria = 'entropy'
				MinSamples = 10

4	0.32	MaxDepth = 4
		Criteria = 'entropy'
		MinSamples = 10
			0.68	MaxDepth = 24
				Criteria = 'entropy'
				MinSamples = 10

results confirm **H2**. See Table 5 for results.

Task Accuracy for Binary Classification. We followed a similar process of analysis for binary classification and ranking tasks. The Friedman Test for Repeated-Measures with post-hoc Wilcoxon signed-rank tests with Bonferroni correction on model accuracy could not prove the statistical significance ($p = 0.73$ for binary classification and $p = 1.03$ for binary ranking task) across SMS and MMS. Similarly we did not observe statistical significance on perceived accuracy ($p = 0.98$ for binary classification and $p = 1.43$ for binary ranking task). Thus, we cannot conclude that MMS outperformed SMS with respect to model accuracy for binary classification and ranking tasks. These results do not confirm **H3** (see Table 4).

Model Switching Behavior: For all participants, the MMS technique resulted in model switching. For participants using the Movies dataset (multi-class classification task) the *max-depth* hyperparameter changed values (ranging from from 3 to 18). Similarly, for the Cities dataset (multi-class classification task) the hyperparameter *Criteria* ranged from *entropy* to *gini*. The *min-samples* hyperparameter varied within the range of 5 to 36 for both datasets. For the binary classification task, *max-depth* ranged from 4 to 9 for both datasets. Also we noticed the *criteria* hyperparameter switching from *gini* to *entropy* for both datasets for the binary classification task.

On average the hyperparameters switched $M = 9.34$ $[7.49, 11.19]$ times to support the multi-class classification and ranking task, while the average change was $M = 5.41$ $[4.89, 5.93]$ for binary classification and ranking task. On the other hand, SMS adhered to the pre-defined hyperparameter setting in each iteration. Though on certain iterations SMS was able to satisfy the majority of the user constraints, however, more frequently SMS failed to satisfy most of the user constraints. For example in the classification task in SMS, the model showed very low cross-validation score (higher is better, see Table 2). This is explainable from the fact that the pre-defined hyperparameter settings in SMS could model the correct decision boundary only on some iterations (as observed from the log-data). On the other hand, for each iteration the MMS technique searched for an optimal model which best characterized the decision boundary of the data. Thus, the decision boundary changed per iteration as the user provided new examples.

Model Performance over time: We compare the model performance in terms of the number of wrongly predicted labels for interacted data instances per iteration. For the multi-class classification task, SMS model output was inconsistent which meant on some iterations the model correctly predicted most of the data items, but the prediction quality may drop in later iterations (see Fig. 7). In comparison to SMS, MMS showed a more consistent performance gain, meaning the number of correctly predicted labels improved over time. However, for the binary classification task, the model performance for SMS and MMS were comparable (see Fig. 7). The results indicate that binary class labels were relatively easier to predict for SMS even if the underlying model's decision boundary was not the best representation of the actual decision boundary of the data instances.

Number of iterations: Using the SMS technique, participants iterated more (compared to the MMS technique) until they found an acceptable ML model. For example, the average number of iterations and standard deviation for the multi-class classification task on the Movies dataset was $M = 3.85$ $[3.31, 3.49]$ with MMS, compared to $M = 6.36$ $[5.08, 7.64]$ with SMS. Thus compared to MMS, users had to iterate more with SMS to adjust the model behavior by showing meaningful examples to suit their goals.

Table 3. The mean, SD, and p-value for all tasks combined for both SMS and MMS. All p-values are Bonferroni-corrected.

Tasks	Single-Model (SMS)	Multi-Model (MMS)	p-val

Model Accuracy All tasks	M = 0.512
	SD = 0.035
		M = 0.786
		SD = 0.054
			< 0.05

Perceived Accuracy All Tasks	M = 0.410
	SD = 0.004
		M = 0.721
		SD = 0.013
			< 0.05

Table 4. Mean, SD, and p-values of model accuracy for both classification and ranking using SMS and MMS. All p-values are Bonferroni-corrected.

Tasks (Model Accuracy)	Single-Model (SMS)	Multi-Model (MMS)	p-val

Multi-class classification	M = 0.623
	SD = 0.005
		M = 0.824
		SD = 0.003
			< 0.05

Multi-class ranking	M = 0.781
	SD = 0.040
		M = 0.912
		SD = 0.023
			< 0.05

Binary classification	M = 0.725
	SD = 0.120
		M = 0.810
		SD = 0.076
			$= 0.73$

Binary ranking	M = = 0.81
	SD = 0.034
		M = 0.832
		SD = 0.233
			$= 1.03$

User Preference: We analyzed the data to for correlations between user preference (Likert scale rating between 1 to 5) per condition (SMS or MMS) with classifier type (multi-class and binary, hot encoded to convert to numeric data, 0 for multi-class, 1 for binary). We found a significant Spearman's correlation (r_s = 0.51, p ¡ .001) between classifier type (multi-class and binary) and system preference (preference of SMS or MMS as indicated by the user). This means that users strongly preferred MMS (Likert rating, $MMS = 4.1$ and $SMS = 3.2$) over SMS for multi-class classification task. For binary task users gave average rating to both systems (Likert rating, MMS = 4.1 and SMS = 3.2). The mean preference for SMS and MMS interface, were 3 and 4.05 respectively (from a score between 1-5). See Figure 9.

7.6 Participant Feedback

Drag and drop interaction: All the participants liked the drag and drop interaction to demonstrate examples to the system. *"I like the drag items feature, it feels very natural to move data items around showing the system quickly what I want"* (P8). However, with a long list of items in one class, can become difficult to move single items. One participant suggested, *"I would prefer to select a bunch of data all at once and then drag-drop them as a collection"*.

Table 5. Mean, SD, and p-values of perceived accuracy for classification and ranking using SMS and MMS. All p-values are Bonferroni-corrected.

Tasks (Perceived Accuracy)	Single-Model (SMS)	Multi-Model (MMS)	p-val

Multi-class classification	M = 0.324
	SD = 0.121
		M = 0.611
		SD = 0.100
			$p < 0.05$

Multi-class ranking	M = 0.418
	SD = 0.055
		M = 0.578
		SD = 0.210
			$p < 0.05$

Binary classification	M = 0.592
	SD = 0.101
		M = 0.622
		SD = 0.051
			$= 0.98$

Binary ranking	M = 0.583
	SD = 0.087
		M = 0.654
		SD = 0.212
			$= 1.43$

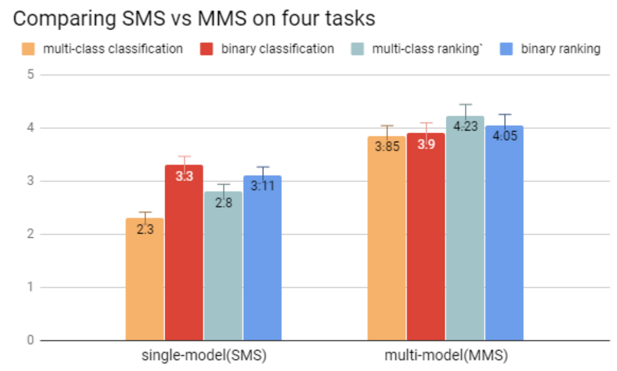


Fig. 9. User preference for SMS vs MMS for the four tasks.

Need for Filtering: Users found the large list of data attributes cumbersome to navigate. P9 said *"I was comfortable with the movies dataset as the number of attributes to look at was less, however for the other dataset (cities), there were way too many attributes to look through and I ended up classifying and ranking by my prior knowledge instead."* Further, participants found the design of the attribute viewer helpful to find representative data items to label. *"Seeing the attributes with red, yellow, and green color encoding helped me understand movies which are high or poorly rated and I made the judgment on popular and unpopular movies based on that"* (P14). However, others pointed that filtering the data by attributes and working on the subset would have helped them find and specify examples to the system more easily. For example, P19 said *"Setting a filter range by attribute would have shown me all the cities with English speaking population with an active nightlife. However, I had to figure that out by hovering over each city one by one."* One of the motivations of the current design was to encourage users to think at the data item level, however filtering functionality may improve performance by reducing the dataset into smaller subsets to work on.

Ease of system use: Our system design methodology was to shield the complexity of model building and model selection as much as possible. We focused on designing the system in a way that encourages users to think about their prior knowledge and communicate that to the system. Most participants found the system simple to use. P12 said *"The process is very fluid and interactive. It is simple and easy to learn quickly."* P12 added *"While the topic of classification and ranking models is new to me, I find the workflow and the interaction technique very easy to follow. I can relate to the use case and see how it [Gaggle] can help me explore data in such scenarios."*

Recommended items: Recommending data while dragging items into various labels helped users find correct data items to label. P12 said *"I liked the recommendation feature, which most of the time was accurate to my expectation. However, I would expect something like that for ranking also ...."* P02 added *"I found many examples from the recommendation panel and I did not have to scroll down. I felt it was intelligent to adapt to my already shown examples."*

User-defined Constraints: The interacted row visualization helped users understand the constraints they placed on the classification and ranking models. P14 said *"The interacted row visualization shows me clearly what constraints are met and what did not. I can simply keep track of the number of blue encodings to know how many are correctly predicted"*. Even though the green highlights in the data viewer also mark the interacted data items, the interacted row view shows a list of all correct or incorrect matches in terms of classification and ranking. P03 remarked *"Without the interacted row view, I cannot keep track of all the data items I interacted with, especially after iteration 4 when the number of items I interacted was over 20."*

8 Discussion and Limitations

Several important challenges and opportunities arose throughout the development and design of Gaggle. We reflect on these below.

Over-fitted Models: One of the challenges in a system like Gaggle is overfitting. An overly aggressive search through the model space might lead to a model which best serves the user's added constraints, but might underperform on an unseen dataset. While we designed Gaggle with the assumption that the data does not necessarily have ground truth in it, understanding the role of overfitting in this context is still and open challenge for human-in-the-loop visual analytic systems.

Large Search Space: Searching models by combining different learning algorithms and hyperparameters leads to an extremely large search space. As a result, a small set of constraints on the search process would not be able to sufficiently reduce the space, leading to a large number of sub-constrained and ill-defined solutions. This leads to the question, how many interactions are considered optimal for a given model space? In this work, we approached this challenge by using Bayesian optimization model ranking. However, for larger search spaces scalability may become an issue. Similarly, too many user constraints may "over-constrain" models, leading to poor results.

Model Selection Strategy: The current model selection strategy in Gaggle automatically picks an optimal model from the pool of n models based on model performance metrics. This shields the user from reviewing each model's complex model metrics and parameters. However, for advanced users we can think of a scenario where Gaggle, instead of selecting the best model, exposes k models for the user to review and select. To ensure a diverse set of models, these k models can be selected such that their output on the data is noticeable different from each other. The intent in such a design will be to give the user a diverse set of models to review and explore.

Abrupt Model and Result Changes: As users steer different models while interacting with Gaggle, there may be significant changes in the results of these models. For example, users might choose Random Forest with "criteria = gini with depth of tree = 50" in one timestep and "criteria = entropy with depth of tree = 2", a model with entirely different hyper-parameter values, and potentially different results. One argument against the possibility of a cognitive dissonance is that, Gaggle was designed with the notion, that in a model, the user only cares about its implication on the data. As the user is not conversant with the meaning of specific model details like parameters or hyperparameters, the design focus should be on the ability to evaluate the efficacy of a model at the data instance level (i.e., if a critical data instance is correctly classified or ranked). The results of these new models are likely judged based on how well they meet the constraints based on the data points (e.g., did the model correctly classify and rank the points I interacted in the previous iteration?). However, exploring methods for showing users what changed between model iterations is an open challenge.

Experimental Design: Our study showed that MMS outperformed SMS in complex multi-class classification and ranking tasks. However, the performance of SMS relies on the choice of learning algorithms and hyperparameters selected. If an optimal single model is selected, the performance of SMS will increase, and may change the results of the study. However, MMS approaches are still valuable in situations where choosing the optimal hyperparameter settings a priori is difficult, or where the user's task may change during the analysis.

9 Conclusion

In this paper, we present a multi-model steering approach for helping people perform classification and ranking tasks. Steering models is a commonly used technique for helping people explore data through the lens of models. Existing visual analytic techniques that leverage model steering focus on steering a single model. However, as exploration typically involves exploring different alternatives and performing different tasks, the need to steer different models becomes important. In response, we present Gaggle, which lets users incrementally steer model hyperparameters of multiple models based on user feedback.

Acknowledgments

Support for the research is partially provided by DARPA FA8750-17-2-0107. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

References

[1] Cars dataset, howpublished = `http://courses.washington.edu/hcde511/s14/datasets/cars.xls`, note = Accessed: 2018-12-11.

[2] Cities dataset, howpublished = `http://graphics.cs.wisc.edu/vis/explainers/data.html`, note = Accessed: 2018-12-11.

[3] Movies dataset, howpublished = `https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset#movie_metadata.csv`, note = Accessed: 2018-12-11.

[4] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.

[5] S. Amershi, J. Fogarty, A. Kapoor, and D. S. Tan. Effective end-user interaction with machine learning. In *AAAI*, 2011.

[6] S. Amershi, J. Fogarty, and D. Weld. Regroup: Interactive machine learning for on-demand group creation in social networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pp. 21–30. ACM, New York, NY, USA, 2012. doi: 10.1145/2207676.2207680

[7] J. Bergstra, D. Yamins, and D. D. Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*, pp. 13–20, 2013.

[8] J. Bernard, M. Zeppelzauer, M. Sedlmair, and W. Aigner. A Unified Process for Visual-Interactive Labeling. In M. Sedlmair and C. Tominski, eds., *EuroVis Workshop on Visual Analytics (EuroVA)*. The Eurographics Association, 2017. doi: 10.2312/eurova.20171123

[9] J. Bernard, M. Zeppelzauer, M. Sedlmair, and W. Aigner. Vial: a unified process for visual interactive labeling. *The Visual Computer*, Mar 2018. doi: 10.1007/s00371-018-1500-3

[10] F. Bouali, A. Guettala, and G. Venturini. Vizassist: An interactive user assistant for visual data mining. *Vis. Comput.*, 32(11):1447–1463, Nov. 2016. doi: 10.1007/s00371-015-1132-9

[11] L. Bradel, C. North, L. House, and S. Leman. Multi-model semantic interaction for text analytics. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 163–172, Oct 2014. doi: 10.1109/VAST.2014.7042492

[12] E. Brown, Y. Sriram, K. Cook, R. Chang, and A. Endert. ModelSpace: Visualizing the Trails of Data Models in Visual Analytics Systems, 2018.

[13] E. T. Brown, J. Liu, C. E. Brodley, and R. Chang. Dis-function: Learning distance functions interactively. In *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, VAST '12, pp. 83–92. IEEE Computer Society, Washington, DC, USA, 2012. doi: 10.1109/VAST.2012.6400486

[14] D. Cashman, S. R. Humayoun, F. Heimerl, K. Park, S. Das, J. Thompson, B. Saket, A. Mosca, J. T. Stasko, A. Endert, M. Gleicher, and R. Chang. Visual analytics for automated model discovery. *CoRR*, abs/1809.10782, 2018.

[15] M. Cavallo and . Demiralp. Clustrophile 2: Guided visual clustering analysis. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018. doi: 10.1109/TVCG.2018.2864477

[16] R. J. Crouser, L. Franklin, A. Endert, and K. Cook. Toward theoretical techniques for measuring the use of human effort in visual analytic systems. *IEEE transactions on visualization and computer graphics*, 23(1):121–130, 2017.

[17] Das, C. Subhajit, C. Dylan, Remco, Endert, and Alex. Beames: Interactive multi-model steering, selection, and inspection for regression tasks. In *IEEE CGA*, 2019.

[18] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, Dec. 2006.

[19] A. Endert, L. Bradel, and C. North. Beyond control panels: Direct manipulation for visual analytics. *IEEE Computer Graphics and Applications*, 33(4):6–13, July 2013. doi: 10.1109/MCG.2013.53

[20] A. Endert, L. Bradel, and C. North. Beyond Control Panels: Direct Manipulation for Visual Analytics. *IEEE Computer Graphics and Applications*, 33(4):6–13, 2013. doi: 10.1109/MCG.2013.53

[21] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pp. 473–482. ACM, New York, NY, USA, 2012. doi: 10.1145/2207676.2207741

[22] A. Endert, C. Han, D. Maiti, L. House, S. C. Leman, and C. North. Observation-level Interaction with Statistical Models for Visual Analytics. In *IEEE VAST*, pp. 121–130, 2011.

[23] A. Endert, W. Ribarsky, C. Turkay, B. Wong, I. Nabney, I. D. Blanco, and F. Rossi. The state of the art in integrating machine learning into visual analytics. In *Computer Graphics Forum*, vol. 36, pp. 458–486. Wiley Online Library, 2017.

[24] 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*, pp. 2962–2970, 2015.

[25] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit. Lineup: Visual analysis of multi-attribute rankings. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, 19(12):2277–2286, 2013. doi: 10.1109/TVCG.2013.173

[26] A. Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, Jun 2016. doi: 10.1007/s40708-016-0042-6

[27] A. Holzinger, M. Plass, K. Holzinger, G. C. Crişan, C.-M. Pintea, and V. Palade. Towards interactive machine learning (iml): Applying ant colony algorithms to solve the traveling salesman problem with the human-in-the-loop approach. vol. LNCS-9817 of *Availability, Reliability, and Security in Information Systems*, pp. 81–95. Springer International Publishing, Aug 2016.

[28] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang. ipca: An interactive system for pca-based visual analytics. In *Computer Graphics Forum*, vol. 28, pp. 767–774. Wiley Online Library, 2009.

[29] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pp. 133–142. ACM, New York, NY, USA, 2002. doi: 10.1145/775047.775067

[30] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *ACM Human Factors in Computing Systems (CHI)*, 2011.

[31] H. Kim, J. Choo, H. Park, and A. Endert. Interaxis: Steering scatterplot axes via observation-level interaction. *IEEE transactions on visualization and computer graphics*, 22(1):131–140, 2016.

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

[33] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, 17:1–5, 2016.

[34] B. C. Kwon, H. Kim, E. Wall, J. Choo, H. Park, and A. Endert. Axisketcher: Interactive nonlinear axis mapping of visualizations through user drawings. *IEEE Trans. Vis. Comput. Graph.*, 23(1):221–230, 2017.

[35] S. C. Leman, L. House, D. Maiti, A. Endert, and C. North. Visual to parametric interaction (v2pi). *PloS one*, 8(3):e50474, 2013.

[36] K. Matkovic, D. Gracanin, M. Jelovic, and H. Hauser. Interactive visual steering rapid visual prototyping of a common rail injection system. *IEEE Transactions on Visualization and Computer Graphics*, 14(6), Nov. 2008.

[37] S. Pajer, M. Streit, T. Torsney-Weir, F. Spechtenhauser, T. Möller, and H. Piringer. Weightlifter: Visual weight space exploration for multi-criteria decision making. *IEEE Transactions on Visualization and Computer Graphics*, October 2016.

[38] S. Paparizos, J. M. Patel, and H. Jagadish. Sigopt: Using schema to optimize xml query processing. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pp. 1456–1460. IEEE, 2007.

[39] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. Boa: The bayesian optimization algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, GECCO'99, pp. 525–532. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[40] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *J. Mach. Learn. Res.*, 11:1297–1322, Aug. 2010.

[41] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):61–70, Jan 2017. doi: 10.1109/TVCG.2016.2598828

[42] D. Ren, T. Hllerer, and X. Yuan. ivisdesigner: Expressive interactive design of information visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2092–2101, Dec 2014. doi: 10.1109/TVCG.2014.2346291

[43] D. Sacha, M. Sedlmair, L. Zhang, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim. What you see is what you can change: Human-centered machine learning by interactive visualization. *Neurocomputing*, 268:164–175, 2017.

[44] B. Saket, H. Kim, E. T. Brown, and A. Endert. Visualization by demonstra-

tion: An interaction paradigm for visual data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):331–340, Jan 2017. doi: 10.1109/TVCG.2016.2598839

[45] M. C. Schatz, A. M. Phillippy, B. Shneiderman, and S. L. Salzberg. Hawkeye: an interactive visual analytics tool for genome assemblies. *Genome Biology*, 8(3):R34, Mar 2007. doi: 10.1186/gb-2007-8-3-r34

[46] Scitkit-Learn. Scikit-learn random search hyperparameter tuning.

[47] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Mller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2161–2170, Dec 2014. doi: 10.1109/TVCG.2014.2346321

[48] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds., *Advances in Neural Information Processing Systems 25*, pp. 2951–2959. Curran Associates, Inc., 2012.

[49] S. Stumpf, V. Rajaram, L. Li, M. Burnett, T. Dietterich, E. Sullivan, R. Drummond, and J. Herlocker. Toward harnessing user feedback for machine learning. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*, IUI '07, pp. 82–91. ACM, New York, NY, USA, 2007. doi: 10.1145/1216295.1216316

[50] S. Stumpf, V. Rajaram, L. Li, W.-K. Wong, M. Burnett, T. Dietterich, E. Sullivan, and J. Herlocker. Interacting meaningfully with machine learning systems: Three experiments. *Int. J. Hum.-Comput. Stud.*, 67(8):639–662, Aug. 2009. doi: 10.1016/j.ijhcs.2009.03.004

[51] 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*, IUI '17, pp. 523–534. ACM, New York, NY, USA, 2017. doi: 10.1145/3025171.3025208

[52] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, Dec. 2009.

[53] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 847–855. ACM, 2013.

[54] L. Tianyi, G. Convertino, W. Wang, and H. Most. Hypertuner: Visual analytics for hyperparameter tuning by professionals. *Machine Learning from User Interaction for Visualization and Analytics, IEEE VIS 2018*, 2018.

[55] S. Van Den Elzen and J. J. van Wijk. Baobabview: Interactive construction and analysis of decision trees. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pp. 151–160. IEEE, 2011.

[56] R. van Liere, J. D. Mulder, and J. J. van Wijk. Computational steering. *Future Generation Computer Systems*, 12(5):441 – 450, 1997. HPCN96. doi: 10.1016/S0167-739X(96)00029-5

[57] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

[58] E. Wall, S. Das, R. Chawla, B. Kalidindi, E. T. Brown, and A. Endert. Podium: Ranking data using mixed-initiative visual analytics. *IEEE Transactions on Visualization Computer Graphics*, 24(1):288–297, Jan. 2018. doi: 10.1109/TVCG.2017.2745078

[59] J. Wenskovitch and C. North. Observation-level interaction with clustering and dimension reduction algorithms. In *Proceedings of the 2Nd Workshop on Human-In-the-Loop Data Analytics*, HILDA'17, pp. 14:1–14:6. ACM, New York, NY, USA, 2017. doi: 10.1145/3077257.3077259

[60] S. M. Yimam, C. Biemann, L. Majnaric, Š. Šabanović, and A. Holzinger. Interactive and iterative annotation for biomedical entity recognition. In Y. Guo, K. Friston, F. Aldo, S. Hill, and H. Peng, eds., *Brain Informatics and Health*, pp. 347–357. Springer International Publishing, Cham, 2015.

[61] Y. Zhou and G. Qiu. Random forest for label ranking. *CoRR*, abs/1608.07710, 2016.