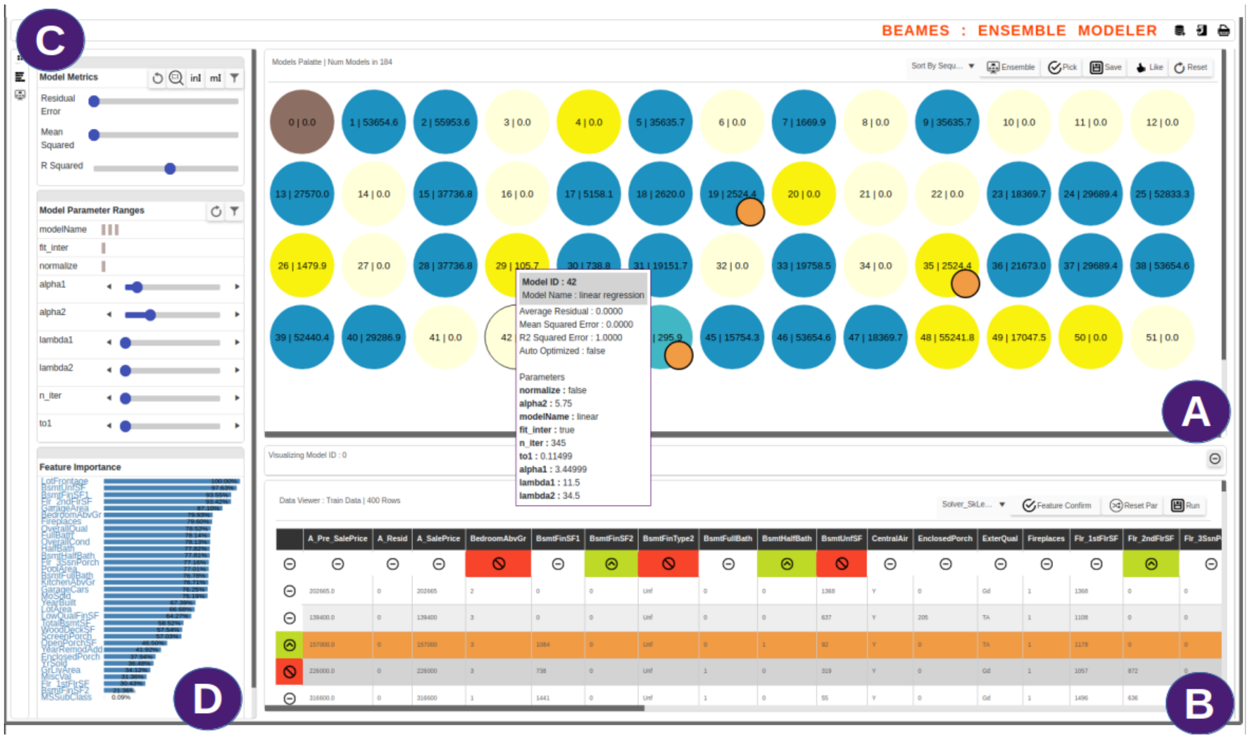
BEAMES: Interactive Multi-Model Steering, Selection, and Inspection for Regression Tasks

Subhajit Das, Dylan Cashman, Remco Chang, Alex Endert



Model ID : 42
Model Name : linear regression
Average Residual : 0.0000
Mean Squared Error : 0.0000
R2 Squared Error : 1.0000
Auto Optimized : false

Parameters
normalize : false
alpha2 : 5.75
modelName : linear
fit_inter : true
n_iter : 345
to1 : 0.11499
alpha1 : 3.44999
lambda1 : 11.5
lambda2 : 34.5

Fig. 1. The BEAMES user interface for multi-model steering, selection, and inspection for regression tasks. The model view (A) shows circular glyphs representing regression models color coded by residual error. The data table (B) shows training, test, and application data sets. The control panel (C) allows users to filter models and critical instances, and change feature weights (D).

Abstract— Interactive model steering helps people incrementally build machine learning models that are tailored to their domain and task. Existing visual analytic tools allow people to steer a single model (e.g., assignment attribute weights used by a dimension reduction model). However, the choice of model is critical in such situations. What if the model chosen is sub-optimal for the task, dataset, or question being asked? What if instead of parameterizing and steering this model, a different model provides a better fit? This paper presents a technique to allow users to inspect and steer multiple machine learning models. The technique steers and samples models from a broader set of learning algorithms and model types. We incorporate this technique into a visual analytic prototype, BEAMES, that allows users to perform regression tasks via multi-model steering. This paper demonstrates the effectiveness of BEAMES via a use case, and discusses broader implications for multi-model steering.

Index Terms—Interactive machine learning, ensemble model builder, multi-model steering, information visualization

1 Introduction

Domain experts use interactive visual analytic systems to solve real problems spanning a broad set of domains. Many such systems incorporate machine learning models to help analyze the data. Users interact with such systems to explore their data by changing various parameters of the models, which in turn produce different results. This process, generally referred to as *model steering*, is complex. Users need to understand the parameters of the models (and the general characteristics of the models) to properly convey their intention or domain expertise to the system, improve the models, and in turn gain insight.

Prior work has looked at this fundamental usability problem. For example, Endert et al. proposed *semantic interaction* as a method to couple model steering operations with native user interaction on the data. Interaction such as highlighting phrases or text, performing a search, or grouping documents steer clustering and natural language processing algorithms [17]. Daee et al. showed with a user study user feedback on feature relevance enhanced sparse linear regression models in a sentiment analysis task [14]. Yang et al. studied design implications for such system, where non-experts can interact with complex models to solve real life problems [60]. Other similar research

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

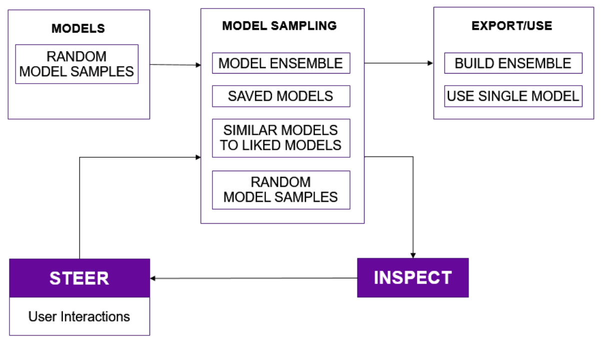


Fig. 2. A conceptual diagram of our iterative technique for multi-model steering and inspection. Users interactively inspect and steer models, which are generated and sampled using a variety of techniques described in Section 3. When satisfied, users can export either a model ensemble or a single model.

explorations can be referred in these works [1,3,11,44].

While the advances of these works are impactful, the complexity of algorithmic support in visual analytic systems and machine learning continue to increase. The act of model steering is no longer limited to adjusting the parameters of a single algorithm. For example, systems like Interaxis [29], Dis-Function [8], AxisSketcher [35], and others rely on interactive updating of a loss function based on user interaction to find optimal attribute weights of a single model that closely matches the domain expertise of the users. These single-model steering systems allow users to intuitively steer a model, without requiring knowledge of the underlying model and its parameters (e.g., [8,19,29,31,57,58]). However, single-model steering becomes less effective when the model chosen no longer fits the task or data characteristics. An incorrectly chosen model is hard to steer to get acceptable results. We call this problem *multi-model steering*, where interaction steers multiple models from a set of model types, and users ultimately select a best model for the task and domain.

Multi-model steering is a complex process requiring a considerable amount of technical expertise. A model is defined by a learning algorithm. Each algorithm relies on a set of hyperparameters. While a model trains on a dataset, it learns an optimal set of parameters and weights to precisely characterize the structure of the data so that it can generalize well on an unseen dataset. Setting the correct learning algorithm and the right combination of hyperparameter values is critical to building an optimal model for a given problem type (i.e., classification, regression, clustering, etc.). For users without formal data science training, specifying each of these parameters may be difficult.

In this paper, we describe a technique that allows users to inspect model outputs, give feedback, and in turn steer and select from multiple models. (See Figure 2). In this case, multiple models refers to a set of models formed by using different learning algorithms, where each algorithm is defined by using a range of values for its hyperparameters (e.g., Model 1 is LogisticRegression(alpha = 0.2), Model 2 is LogisticRegression(alpha = 5), Model 3 is BayesianRegression(alpha = 40), etc.).

The visual analytics technique presented in this paper allows domain experts to inspect models by checking a model's predicted output on the data (i.e., checking critical data instances). Accurate prediction of critical data instances can increase user's trust in the model. Our technique also allows people to steer and inspect multiple models. Further, our technique assists the inspection process by recommending models (from the collection of models) which successfully make predictions on the critical data instances with zero or relatively low error value. Showing a wide spectrum of models for the given regression problem can be beneficial to domain experts who otherwise would not be aware of the many possibilities and permutations of models. Being able to filter the data by instances and filter models by their performance (by simple double range sliders and toggling switches for categorical items), users can drill down to models which are successful and can validate them by checking their results on critical data instances. However, since this process is iterative, our technique provides an interactive visual interface to iteratively refine the critical instances and other user input to continue model steering and inspection. Further, our technique enables domain experts to add knowledge to the model building process. Users can add knowledge about which features may be more important than others, or which data instances are more important to correctly predict.

The technique presented in this paper has three primary components: i) interactive weighting of critical data instances, ii) interactive feature selection with weights, and iii) interactive model selection and iv) building model ensembles. Users can steer multiple models simultaneously by increasing the weights on critical data instances (using the on-demand sliders shown in Figure 3), indicating that accuracy on these data items is more important. In addition, they can select features (by toggling checkbox type buttons) and specify their weights (by dragging sliders) to specify their relative importance. Users can also perform interactive model selection by "liking" one or more models, from which BEAMES generates a new set of similar models to inspect. Finally, they can select a export a model that best fits their task, or generate an ensemble of models to use.

To summarize, our technique searches the model space for models that more closely adhere to data items and attributes the user is interested in. The set of models is refined at each step, where more models are generated and less relevant models are pruned. This human-in-the-loop process allows domain experts to explore a myriad of models for their regression task, and add domain expertise into the model building to produce models which adhere to both subjective and objective preference of the users. The main contribution of this paper are:

1. An interactive technique for domain experts to select an optimal model using multi-model steering aided by - (1) interactive weighting of data instances; (2) interactive feature selection and weighting; and (3) building model ensembles.

2. A visual analytic system called BEAMES, which instantiates our technique to help people select a regression model using multi-model steering.

3. A usage scenario demonstrating the intended usage and interaction with the system.

2 Related Work

2.1 Single Model Steering Systems

Interaction based single model driven visual analytic systems has been around for a while, helping non technical users build and change model parameters by control panels or UI elements which enables them interactively demonstrate feedback. The spectrum of problem types these systems solve is adequately wide. It includes ranking [57], metric learning [8], decision trees [55], dimensional reduction [19,29,35], feature selection [25], weight space exploration [38] and many more. For instance, Podium [57], is driven by a single linear SVM model with the goal to compute attribute weights based on users subjective preference of multi attribute data items. Further demonstration-based interaction using a single model steering approach include systems such as [8,16,18,19,36].

In all of these examples the model infers parameters based on users demonstration by direct manipulation of graphical widgets. A relevant system is the work of Kim et al., InterAxis [29], 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. Mühlbacher et al. [37] explains increased user involvement in black box algorithms, using parameter refinement to change the underlying models. Pezzotti et al. [41] have shown a single user steerable model to provide feedback to tSNE models for dimensionality reduction. Similarly, in an interactive recommender system, a user can provide continuous feedback, such as by recording additional choices, or by explicitly scoring (liking/disliking) individual items [24]. Many other examples of direct manipulation of visual glyphs to provide user feedback exist [5, 18, 28, 45, 48, 51, 52, 58]. Our work is distinct from these as we are enabling the user to steer (by interaction based user feedback) multiple machine learning models as opposed to single models. Also, our technique allows users to inspect multiple models simultaneously, leveraging them to evaluate and select an optimal model.

Liere et al. have defined computational steering as a process to enable users to change parameters of simulations on the fly [56]. Their paper emphasizes the concept that simulations run over many iterations, where users may need to update parameters before completion. In this paper, we ground our concept of model steering in this prior work, and refer to it as a process in which a model's parameters are changed to iteratively produce updated results, and multi-model steering as a process in which a model's hyperparameters and parameters are changed. It is similar to computational steering in that multiple iterative cycles are computed, and user input can change parameters or hyperparameters at any iteration. This also aligns well with how model steering has been implemented in visual analytic systems discussed above. Additionally, BEAMES also supports interactive model selection by allowed users to select a subset of models and generate a new set of similar models.

2.2 Multi Model Interaction and Model Ensembles

Work of Bradel et al. [6] lies in the multi model steering space. Using semantic interaction they allow users to steer multiple text analytic models. While effective, their system is scoped to text analytics and handling text corpora at multiple levels of scale. Shneider et al. showed visual integration of data and model space, by allowing users identify effective component models on data items from a classification model ensemble [49]. Patel et al. showed an example technique to work with multiple model systems helping users understand relationship between data, models and features [40]. Piringer et al. showed an interactive visual analytic system helping multiple regression model comparison and validation in an interactive fashion [42]. Their technique specifically uses comparison of multiple model outputs to help users select the best model. Similarly, Cutura et al. prototyped an interactive multi-model selection tool focused on the comparison of multiple dimensionality reduction models [46]. Kwon et al. [34] showed a tool to visually identify and select an appropriate cluster model from multiple clustering algorithms and parameter combinations. However, their work targeted data scientists as the user, while we are aiming to build techniques for domain experts without formal data science training.

In the literature, there is ample contribution in model space and parameter space analysis to traverse the model space. For example, to characterize user's role in interactive machine learning space, Amershi et al. [2] proposed a set of high-level paradigms. Sedlmair et al. [50] defined visual parameter analysis as variation of model parameters, generating a diverse range of model outputs for each such combination of parameters. Their work investigated the relationship between the input and the output within the described parameter space. The work presented in this paper specifically focuses on tabular data model building for a regression task by non expert users. More importantly, we are using multiple models to enable users find an optimal model from a broader set of model types.

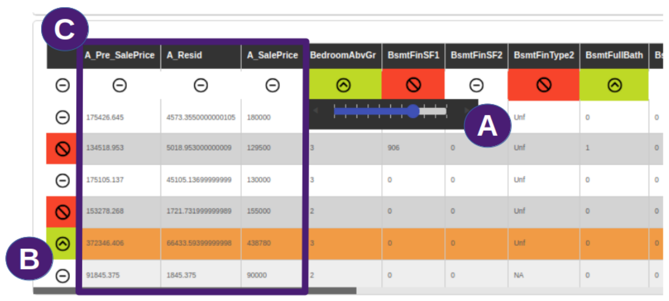
The topic of model ensembles is related to our work, as one of the interactions provided to the user is to build ensemble models from a set of models they like. Model ensembles increases model performance by fusing multiple model's strength. Different strategies yield different kinds of model ensemble [26, 27]. For example, Potter et al. [43] showed an interactive ensemble model to allow focus and discovery of simulation outcomes. Datta et al. built a system CommunityDiff, showing a mechanism to visualize ensemble space by using weighted combination of various algorithms to aid identifying patterns, commonalities and differences in the space of community detection problem type [15]. Talbot et al. [53] built an interactive ensemble matrix system visualizing confusion matrices to allow insight gain on various component classifiers. Model ensembles can be built by training the component models on different subset of data [7, 21], or by using different algorithms [33, 59] for each model type (e.g., by using bagging [7]). In our case, we allow users to select models they like and combine them to build a model ensemble via bagging.

Table 1. Model samples and hyperparameters used by the learning algorithm.

Model	Learning Algorithm	Hyperparameters
M1	Linear Regression	**fitIntercept** = 'true', **normalize** = 'false'
M2	Linear Regression	**fitIntercept** = 'false', **normalize** = 'false'
M3	Linear Regression	**fitIntercept** = 'true', **normalize** = 'true'
M4	Logistic Regression	**fitIntercept** = 'true', **penalty** = 'l1', **dual** = 'true', **tol** = '10.55', **C** = '1.854', **maxIter** = '10'
M5	Logistic Regression	**fitIntercept** = 'false', **penalty** = 'l2', **dual** = 'true', **tol** = '0.12', **C** = '100.854', **maxIter** = '20'
M6	Logistic Regression	**fitIntercept** = 'true', **penalty** = 'l2', **dual** = 'false', **tol** = '-5.32', **C** = '55.4', **maxIter** = '50'
M7	Bayesian Regression	**fitIntercept** = 'true', **normalize** = 'false', **alpha1** = '0.85', **alpha2** = '-5.32', **lamba1** = '55.4', **lambda2** = '50.524', **computeScore** = 'true'
M8	Bayesian Regression	**fitIntercept** = 'false', **normalize** = 'true', **alpha1** = '20.85', **alpha2** = '-51.112', **lamba1** = '155.422', **lambda2** = '-30.24', **computeScore** = 'false'
M9	Bayesian Regression	**fitIntercept** = 'true', **normalize** = 'true', **alpha1** = '8.65', **alpha2** = '1.102', **lamba1** = '-5.45', **lambda2** = '50.24', **computeScore** = 'true'
M9	Ensemble Regressor	**compModels** = '['m1', 'm3', 'm4']', **maxFeatures** = '10', **maxSamples** = '200', **randomState** = '45', **numIter** = '30'
M10	Ensemble Regressor	**compModels** = '['m4', 'm2', 'm8', 'm3']', **maxFeatures** = '500', **maxSamples** = '100', **randomState** = '45', **numIter** = '100'

2.3 Automated Model Selection

Model building is a non-trivial task for non expert users, as it involves complexity including selecting a good combination of learning algorithms and hyperparameters. One solution is to use existing automated model selection tools such as AutoWeka [32, 54], SigOpt [39], HyperOpt [4, 30], or AUTO-SKLEARN [20]. These tools follow numerous optimization procedures internally to find the right combination of algorithms and hyperparameters for an optimal model suited to the given dataset and task. One drawback is that these processes rely on quantifiable or objective features in the data. Subjective nuances or extra knowledge from the users is not incorporated in typical automatic model selection pipelines. Instead, our technique does not follow an auto-optimization based process to select an optimal model, as is the case in many of these auto machine learning (auto-ML) workflows. Sacha et al. present a comprehensive framework for a human-centered machine learning workflow [47]. They describe the interaction and relationship between automated algorithms and information visualization [47]. BEAMES leverages user feedback on model outputs to incrementally steer and select models, which can be mapped to this comprehensive model.



	A_Pre_SalePrice	A_Resid	A_SalePrice	BedroomAbvGr	BsmtFinSF1	BsmtFinSF2	BsmtFinType2	BsmtFullBath	Bs
⊖	⊖	⊖	⊖	🟩	🟥	⊖	🟥	🟩	
⊖	175426.645	4573.3550000000105	180000				Unf	0	0
🟥	134518.953	5018.953000000009	129500	3	906	0	Unf	1	0
⊖	175105.137	45105.13699999999	130000	3	0	0	Unf	0	0
🟥	153278.268	1721.731999999989	155000	2	0	0	Unf	0	0
🟩	372346.406	66433.59399999998	438780	3	0	0	Unf	0	0
⊖	91845.375	1845.375	90000	2	0	0	NA	0	0

Fig. 3. (A) The data table view showing the toggle switches (green, white and red) for features. The sliders allow users to add weights to the data samples and attributes.(B) Similar toggle switches and sliders for data samples. (C) From left : Column 1 is the predicted output, Column 2 is the residual error, Column 3 is the ground truth value to predict, in this case sale price of a house.

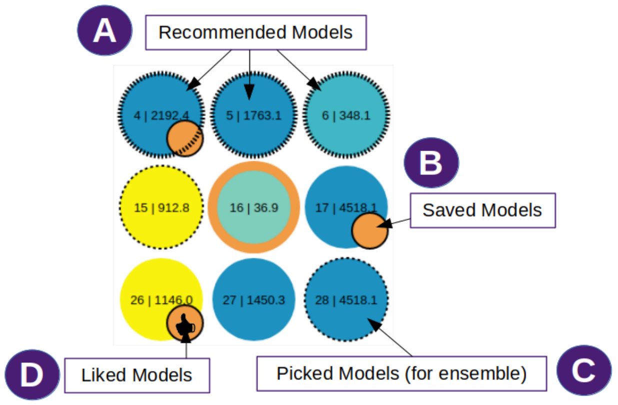


Fig. 4. Shows the model view in detail. (A) Recommended models with strong border stroke. (B) Saved models (C) Picked models for model ensembles (D) Liked models.

3 System Description: BEAMES

This section describes BEAMES, a visual analytic system for multi-model steering to solve a regression task. BEAMES enables users to inspect outputs of multiple models, specify critical data instances, steer multiple models to increase or refine their performance, and finally select one model (or an ensemble of models) when satisfied. It assists the user to select an optimal regression model from a collection of models constructed by various combination of regression algorithms and their hyperparameters. Some example algorithms tested includes Bayesian Regression, Linear Regression, and Logistic Regression (shown in Table 1). In Section 3.1, we describe the main components of the BEAMES user interface. Section 3.2 discusses the underlying techniques implemented to translate user interactions into appropriate model responses.

3.1 User Interface

The user interface (shown in Figure 1) consists of four primary views: a data table, a model view, a control panel, and a model detail view.

3.1.1 Data Table

Users can see the loaded dataset with every attribute in the data table view (Figure 1-B). By default the view shows the training data but users can toggle to view test and application data set (test data required to validate models). We define application data set as the final data set without labels/output values. The view uses a standard spreadsheet metaphor, representing data items as rows and attributes as columns. Users can add, update or delete training data (both rows and columns). The columns (attributes) of the loaded data shows three state toggle switches allowing users to emphasize, de-emphasize, or discard an attribute using a slider to adjust the weights (shown in Figure 3). Likewise, users can discard a data instance or increase the relative importance of critical instances. Increased weighting tells the model that it should learn more from this instance than others, and emphasize the accuracy of these instances more. Users can also add domain expertise to the data via new attributes if wanted. Hovering over any row on the table triggers the system to recommend models from the model view to the user to inspect (i.e., models that correctly predicted the data instance). Recommended models are shown with a border stroke on the circular glyphs (representing a model) in the model view (Figure 5).

3.1.2 Model View

The model view shows each model as a circular glyph as shown in Figure 1-A. They are color coded by residual errors on the data loaded on the data table for the given regression problem. Yellow represents lower residual error, while dark blue represents higher residual error. The text on the glyphs describe the residual error value.

These circular glyphs are interactive. Hovering shows the model output details like the number of instances correctly predicted, the model's learning algorithm, etc. Users can inspect a model by clicking a visual glyph which adds a column to the data table, showing the predicted value (e.g., housing price on a housing data set). The prediction column is next to the ground truth column, allowing users to compare how close the model's prediction is to the ground truth for each data instance. Inspecting a model also opens the model detail view (explained below). Furthermore, users can select multiple models to build model ensembles, or save a single model, to preserve the model across iterations. Users can also like a model, specifying that the next iteration should sample models similar to the one's the user liked (See Figure 4).

3.1.3 Model Detail View

The model detail view shows up when the user inspects a model by clicking on it. It draws a line chart showing the predicted model output and ground truth data (shown as dots, shown in Figure 6-A). Visually it tells users how accurately the model fits the data. In the same view, users can see a residual bar chart, depicting the amount of error in prediction by the inspected model on both training and test set (See Figure 7). The third view (bottom-left) shows correlation between two selected attributes from the data to the user. It helps users understand relationships between attributes in order to know which ones to emphasize, de-emphasize, or discard.

3.1.4 Control Panel

The control panel (see Figure 1-C) contains frequently-used operations, such as filtering data instances and attributes. Data instances can be filtered by quantitative attributes using the double-range sliders. Categorical attributes can be filtered by toggling rectangular glyphs. Similarly, models can be filtered by dragging the sliders which specify a threshold range of model accuracy, residual scores, or desired number of correctly predicted instances.

3.2 Technique

In this section, we will describe the underlying techniques applied to enable multi model inspection and steering.

3.2.1 Data

We define our full data set as C (contains $N+K+B$ instances) which is then split in training, test and application data set. Users train models on the training set D containing N samples, then validate on test set T containing K samples. When they find an acceptable model they export it or use it on application data set H containing B samples. For notation description please refer Table 2.

Table 2. Notation and definitions used to define our technique.

Notation	Description
$C = D \cup T \cup H$	Full Data set, comprising of training, test and application data set
$D = d_1, d_2, ...d_N$	Training data set of size N
$T = t_1, t_2, ...t_K$	Test data set of size K
$H = h_1, h_2, ...h_B$	Application data set of size B
$A = a_1, a_2, ...a_P$	Set of attributes in the data
P	Cardinality of data attributes
$W = w_1, w_2, ...w_N$	Set of attribute weights
$\Omega = \omega_1, \omega_2, ...\omega_P$	Set of training data weights
$M = m_1, m_2, ...m_S$	Set of models of size S
$L = l_1, l_2, ...l_J$	Set of J learning algorithms
$P = p_1, p_2, ...p_J$	Probability distribution to pick J learning algorithm
$\Lambda_k = \lambda_{k,1}, \lambda_{k,2}, ...$	Set of hyperparameters for model k

3.2.2 Model Sampling

We define a *model*, M_i as a function $f : \mathscr{X} \mapsto \mathscr{Y}$, mapping from the input space $\mathscr{X}$ to the prediction space $\mathscr{Y}$. Here, the prediction space is $\mathbb{R}$ and each model m_i is a regression model. For the modeling algorithms we used Scikit Learn's machine learning package [9]. Each model is sampled by combining a learning algorithm l_k from a set of J algorithms (hand picked by us for the regression task). Tested algorithms include Linear Regression, Logistic Regression, and Bayesian Regression. Each learning algorithm comes with their own set of hyperparameters λ_m. Examples of sampled models can be found in table 1. A sampled model is defined as below :

$$m_i \mapsto Model(l_k, [\lambda_{k1}, \lambda_{k2}, \lambda_{k3}....])$$

The system initiates with randomly sampled S models. We'd like the sampling distribution to be uniform across algorithms such that users can inspect a wide spectrum of model outputs for the given regression problem. For that reason, we initialize probability p_k to sample a learning algorithm l_k (for a model m_i) from J possible models as $1/J$.

3.2.3 Updating training data

Users can load training data D on the data table. Every data sample is initially set to an equal weight of $\omega_i = 0.5$. However, users can interactively set weights on the samples between 0 and 1. 0 meaning to discard the data sample in training, while 1 is to place highest strength to the learning from the sample.

$$a_{ij} = \begin{cases} x & \text{user added value to the subset data instances} \\ 0 & \text{initialized value for the rest of the data set} \end{cases} \quad (1)$$

where, a_{ij} represents data at i th column and j th row.

3.2.4 User driven feature engineering

Similar to weighting of data instances, users can emphasize, discard or weight quantitative features. Using the UI toggle buttons (See Figure 3-A), users can specify if they want to emphasize or discard a feature for model training. Discarding a feature removes it from the set **A**. Emphasizing a feature reveals a weight slider, which the user can toggle between -1 and 1. Setting a weight of -1 enforces the model to place higher emphasis on lower values of the attribute than others. As with instance weighting, all attributes default to have user weights of 0.5 before any interaction takes place.

The instance and attribute weights assigned by the user directly affect the computation of the y dependent variable similar to the work by Cleveland [12]. Loss for the models is a weighted least squares loss. Thus, the different regression models solve the following regression problem.

$$\min \sum_{i=0}^{N} \omega_i * (\hat{y}_i - y_i)^2$$

where,

$$\hat{y}_i = b_0 + \sum_{i=0}^{M} b_i * x_i * w_i$$

ω_i is the user defined weights for data instance i, b_0 is the intercept, b_i is the coefficients of the attributes learned by the model, and w_i are the user's attribute weights.

3.2.5 User Interactions

User interactions in BEAMES are designed to update the underlying models via both interactive model steering and selection. This section describes these interactions, and details how the models interpret and update accordingly.

Save Models: If the user saves a model M_i, the system saves its learning algorithm L_i and the set of hyperparameter combination represented as $[\lambda_1, \lambda_2, \lambda_3...\lambda_m]$. At each iteration of model sampling, BEAMES keeps saved models in the set of models shown to the user.

Like Models: If the user toggles the Like button in the interface on model m_d, then the probability p_d of the learning algorithm that produced that model is increased by a factor r_f. We randomly set the value of r_f by using a threshold ε. With trial and error, we found $\varepsilon = 0.1$ showed promising results. Likewise, the hyperparameters λ_i of that algorithm are sampled from within a threshold region of the hyperparameters used in the liked model. This ensures that a large share of the new sampled models are from the neighboring regions of the models users liked. However, the technique still ensures randomly sampling a smaller share of other models in the collection such that the user can still get an overview of model output from a wide array of choices. The process of liking models to regenerate a new set of models that are similar helps users perform interactive model selection.

Adjust Data Instance or Attribute Weight: Users can adjust the weights of data instances and attributes by adjusting the respective sliders. As a result, all the available M models are retrained using N training instances with user specified features A_k where $A_k \subset A$ and user specified feature and data instance weights, W and Ω respectively.

Export Model: Once users are satisfied with a model, they can simply export the single model. Additionally, model ensembles can be created and exported. For instance, users can select (by the pick interaction) G models to build a model ensemble. The system uses each component model m_j to build a model ensemble E. We are following the model ensemble technique described by Caruana et al. [10], where they used a stepwise selection process to select component models to build ensemble which increases ensemble accuracy. Our technique differs from theirs in that BEAMES provides users the flexibility to select component models to build an ensemble.

BEAMES uses a bagging technique [7] to sample from training data (sampling with replacement). However, data instances d_i which the users have increased the weights ω_i get higher probability to be sampled than other instances. This is to make sure models learn from these samples as users have intended, and the ensemble model emphasizes accuracy on the critical data instances d_i. Given the regression problem, the system finds the final predicted output by averaging the output received from component models. However, if any of the component models m_i are saved or liked by the user, then the output computed is weighted, and the weights of the saved or liked models are higher than unsaved models. The final ensemble model E's output is the weighted average of the predictions of the models in the ensemble.

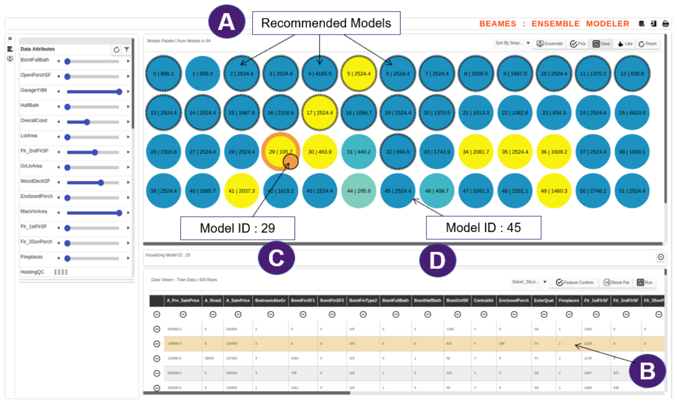


Fig. 5. View showing Amy loads BEAMES to find 64 regression models. (A) Recommended Models (B) Amy hovers over a critical data instance. (C) Amy clicks model 29 and saves it. (D) Amy clicks on model 45 and inspects the output.

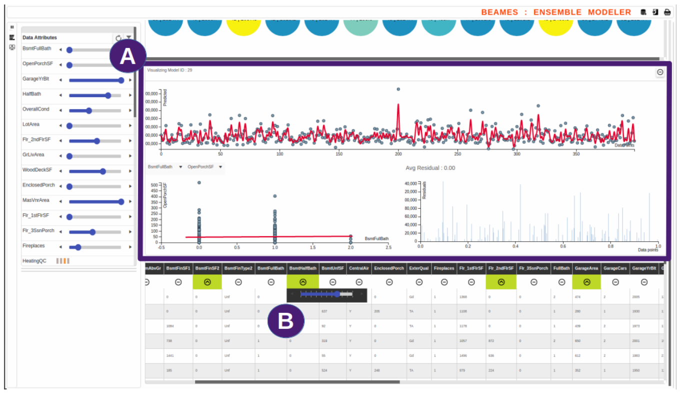


Fig. 6. (A) Shows the model detail view. (B) Amy emphasizes features using the slider.

4 Usage Scenario

We describe our tool BEAMES with a usage scenario, where a domain expert uses the system to perform data exploration and predict future housing prices. Amy is a real estate agent who reviews existing and new properties to analyze their market prices and potential change in the future due to changing conditions in the city. Amy has years of experience in her field including field knowledge of the city's various neighborhoods, upcoming city projects, infrastructure changes, and other city planning activities. She has a good grasp of the changing demographics of the city and the rising demand for housing. She usually explores data using tools like MS Excel,to decide on which properties to buy, sell. However, not being a data scientist she is not conversant with complex modeling techniques, which can help her accurately predict property prices, property demands, or ratings in future.

Amy begins by importing two datasets into BEAMES (the dataset is available here [23]). The first dataset is for the properties she knows the prices of at the current state of the market (called input data), and the second has properties she wants to predict future prices of (called the application data set). BEAMES splits the input data into training and test sets for model training and validation. The training data have over 750 samples with 36 attributes comprised of both categorical and quantitative types. It has a target attribute namely $SalePrice$, containing the property price of each house. Every row in the data is a property(a house) described by attributes such as property size, fireplaces, year built, number of bedrooms, etc. The application data set has over 800 unlabeled samples.

After importing the data, BEAMES builds 64 regression models, each randomly sampled using a combination of learning algorithms (linear, logistic, ridge, and bayesian regression) and hyperparameter values (alpha, lambda, tol etc.). The list of sample models with hyperparameter values is shown in Table 1. As Amy is not formally trained in the specifics of the models, she begins her exploration by how well specific models predict property sale prices. In the model view (Figure 5), she sees the collection of models as circular glyphs color coded by their residual error scores. Yellow represents better models with lower residual error, while blue glyphs are models with high residual errors. On the bottom in the data table, Amy sees the training data in a tabular format. Browsing the colors of the circular glyphs (representing models), Amy decides to start inspecting a few further as they have lower error values.

Clicking the circle, Amy sees that the training data has a new column representing predicted price. She sees that the predictions are quite close to some of the actual prices in the data. She clicks on model 45 (see Figure 5-D), as it has accurately predicted over 300 entries on her training data and has some errors on the rest of the entries from the training data. She sees that the residual error of the current model is over 500. Next, she notices few models with residual error score close to 100 (colored yellow). She clicks one of the lower scored models (model id 29, as seen in Figure 5-C) and finds most of the data instances are predicted accurately. However, to double-check if some of the known data instances were correctly predicted, she uses the filter panel on the left. She sees the currently selected model (with a low residual error of 105.7) did not correctly predict most of the these critical data instances.

By hovering over these critical data instances, (see Figure 5-B) the system shows models that Amy should inspect, as they made correct predictions on those instances (See Figure 5-A). Amy reviews a few of the suggested models. She finds that the recommended models performed better for the critical instances, though they had higher overall residual errors. Next, Amy toggles the three state toggle button on these data instances to emphasize them using the slider (shown in Figure 6-B). In addition, Amy thinks the property price should be most strongly defined by the $numberofbedrooms$, $garageArea$, $2ndfloorarea$ attributes. She again uses the slider to increase their weight, while reducing the weight on $frontPorchSize$ and $DrivewayQuality$. Next she presses the build new model button for BEAMES to recompute all the models.

BEAMES updates the model view with newly computed models. Amy sees the color encoding changed for the collection of models, as new models have different residual error output. Amy quickly looks over the collection to find many models have scores close to 0, meaning they have high performance on training data. However, still being interested in her critical data instances, she hovers over the rows to see recommended models from the model view. She clicks on few of the recommended models and finds two of them perform quite well, as it correctly predicted 7 out of the 8 critical instances. Clicking on these models, Amy finds the prediction on the the test data is bit off. For example, property id 104 shows a predicted price of 141,345, while the true price is 99,322.

Confused to find relatively poor performance on the test set, Amy uses the control panel to see the importance of the features in the horizontal bar chart. She sees the relatively strong weight on the $numberofbedrooms$ attribute (as she intended previously). She adds few other relevant attributes and ups the weight factor for those, i.e, $overallPropertyRating$, $numberOfFloors$, $distanceToTransit$. Next, she discards a few training data samples thinking those properties are not relevant anymore. She saves two models that she found have good potential and likes a few others based on their performance. In addition she picks a few models to create an ensemble from these component models, and generates more models.

Amy browses the newly computed models. She sees brown colored circular glyphs (See Figure 1-c), representing an ensemble model build from the models she picked. She clicks on it and finds that it shows a very accurate prediction on the training data. Amy confirms the same on the model detail view, as the line fits the set of points (representing actual ground truth values), as shown in Figure 7. Similarly, she sees

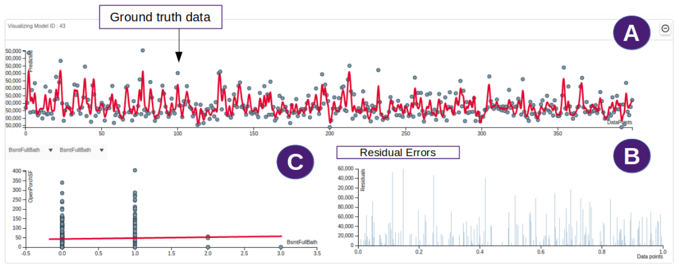
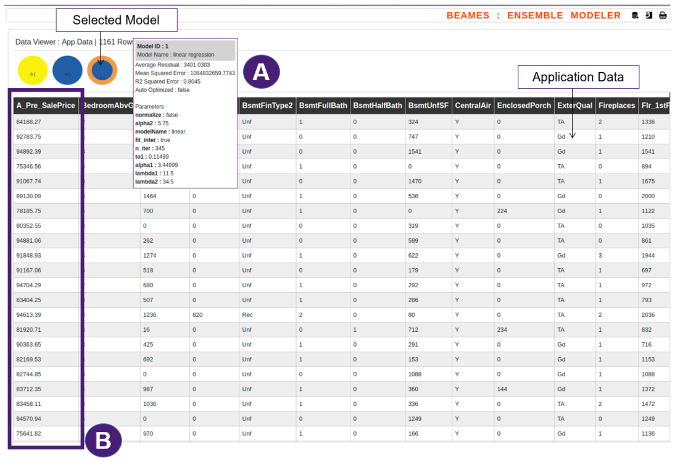


Fig. 7. Amy exploring the model detail view showing low residual error for most of the instances.



Data Viewer : App Data | 1161 Rows

Model ID : 1
Model Name : linear regression
Average Residual : 3401.0303
Mean Squared Error : 1084832659.7743
R2 Squared Error : 0.8045
Auto Optimized : false

Parameters
normalize : false
alpha2 : 5.75
modelName : linear
fit_inter : true
n_iter : 345
to1 : 0.11499
alpha1 : 3.44999
lambda1 : 11.5
lambda2 : 34.5

A_Pre_SalePrice	BedroomAbvG			BsmtFinType2	BsmtFullBath	BsmtHalfBath	BsmtUnfSF	CentralAir	EnclosedPorch	ExterQual	Fireplaces	Flr_1stF
84188.27				Unf	1	0	324	Y	0	TA	2	1336
92783.75				Unf	0	0	747	Y	0	Gd	1	1210
94892.39				Unf	0	0	1541	Y	0	Gd	1	1541
75346.56				Unf	1	0	0	Y	0	TA	0	894
91067.74				Unf	0	0	1470	Y	0	TA	1	1675
89130.09		1464	0	Unf	1	0	536	Y	0	Gd	0	2000
78185.75		700	0	Unf	1	0	0	Y	224	Gd	1	1122
80352.55		0	0	Unf	0	0	319	Y	0	TA	0	1035
94881.06		262	0	Unf	0	0	599	Y	0	TA	0	861
91848.93		1274	0	Unf	1	0	622	Y	0	Gd	3	1944
91167.06		518	0	Unf	0	0	179	Y	0	TA	1	697
94704.29		680	0	Unf	1	0	292	Y	0	TA	1	972
83404.25		507	0	Unf	1	0	286	Y	0	TA	1	793
94613.39		1236	820	Rec	2	0	80	Y	0	TA	2	2036
81920.71		16	0	Unf	0	1	712	Y	234	TA	1	832
90363.65		425	0	Unf	1	0	291	Y	0	Gd	1	716
82169.53		692	0	Unf	1	0	153	Y	0	Gd	1	1153
82744.85		0	0	Unf	0	0	1088	Y	0	Gd	1	1088
83712.35		987	0	Unf	1	0	360	Y	144	Gd	1	1372
83458.11		1036	0	Unf	1	0	336	Y	0	TA	2	1472
94570.94		0	0	Unf	0	0	1249	Y	0	TA	0	1249
75641.82		970	0	Unf	1	0	166	Y	0	Gd	1	1136

Fig. 8. Amy loads the application data set to apply the saved models and see predicted output. (A) Horizontal panel storing models saved by Amy. (B) Predicted output ("sale price") when a model is selected.

almost 0 residual error from the bar chart shown in Figure 7.

At this point, she also uploads the test dataset in the data table. She hovers over the rows, to see system recommended models (visually represented with an outer stroke line). She is happy to see that the models recommended include the ensemble model and the one she saved. She clicks on the recommended models and finds that the results improved. The predictions were very close and in some instances were almost the same as the ground truth. At this point, Amy is happy with the models and saves a few (including the model ensemble). She loads the application data set. The interface changes to show a full-screen spreadsheet, with a horizontal scrollable panel on top listing the saved models (Figure 8-A). She clicks on the saved model glyphs to see predictions on the application dataset. As she clicks, BEAMES adds a prediction column to the application data set (Figure 8-B).

At this point, Amy has models to help her predict house prices, each emphasizing different characteristics of the data. As she continues her work throughout the coming months, she can use these models to give her a sense of how housing prices may change.

5 Discussion

Spectrum of automatic, semiautomatic, and manual model selection. This work in this paper is a semi-automatic technique for exploring multiple models and their respective parameterizations. This technique takes a different approach from automatic hyperparameter tuning model selection, as adopted by some of the existing systems like AutoWeka and Hyperopt. Our technique also differs from manual model-building techniques where users are required to specify most (if not all) of the model parameters. Within this model selection spectrum, we explore how to bring domain experts in the loop of model building by performing model validation and selection based on not only on objective metrics (i.e., residual errors) but also data instance validation (i.e "I like model *A* because it predicts instances 1,5,10 correctly, though it has relatively higher residual error."). We call this approach semi-automatic, because some parts of model sampling and model building are still automatic.

Model Output Inspection and Model Interpretability. Model interpretability and comprehensibility is discussed in the work of Gleicher [22]. Gleicher discusses model interpretability as a means to understand the learning of the model. Craven [13] defined model comprehensibility to be if a model's learning algorithm can be encoded in such a way that it is legible by a human being. However, in our work, we are aiming to help users understand a model by inspecting its output and steering it towards their own sense of the importance of familiar and critical data instances. Our visual technique helps users understand the model output against ground truth, without having to interpret the internal complexities of the models. While our work begins to explore this space, there is more work to understand how users understand, interpret, and trust models by observing outputs.

Incorporating Domain Knowledge. In our technique we aim for a framework which allows domain experts to add knowledge to the model learning process. In BEAMES, there are many ways to do so (i.e., adding a new attribute to a subset of data, discarding data samples, emphasizing weights on critical data instances, adding feature weights, etc.) These interactions directly update every model's underlying weights and parameters to incorporate the user feedback. The output of the updated models can be inspected to understand how they changed. However, there is an opportunity to explore other techniques to inform users how their feedback was incorporated by models, beyond what was explored in this paper.

6 Limitation and Future Work

Model Space Sampling. The number of models that can be sampled for a given problem/task and dataset is large (potentially infinitely so). Given the plethora of options for learning algorithms, each with a distinct group of hyperparameters, which can take values within a set domain range, the size of the model candidate space grows rapidly. Any multi-model optimization technique searching from such a large model space can result in many model to choose from. In BEAMES, we set an upper limit on the number of model options the system randomly samples to initiate the loop of model inspection and steering. Another feature assisting the user is the technique to recommend models to inspect. The recommendations are driven by how well the models perform on data instances the user care about. However, given the iterative nature of the technique, it might lead to a substantial amount of user interaction until they find an acceptable model, leading to fatigue or frustration. In the future, we are interested in exploring additional forms of guidance for multi-model steering.

Beyond Model Export. At the current state, the technique and the interface allow domain experts to find an optimal model and use it to work with a application data set. It can also export the saved models as a locally saved file (JSON format). However, there are many opportunities beyond model export that can be considered. Can users save models and use them across entirely different datasets but similar problem types? Alternatively, can multiple domain experts work on previously saved models and extend them further, based on the changing nature of the domain?

Model Overfitting. Externalizing a domain experts knowledge to train a model can often lead to the problem of overfitting. Though BEAMES has a provision of using a test data set to train model, we understand that the overuse of the testing data might lead to an overfitted model. The current interface of BEAMES does not have explicit functionality to avoid model overfitting. In future work, we look forward to overcoming model overfitting while still using expert knowledge in the model training process.

Model Comparison. BEAMES in its current state can help users compare models by inspecting their outputs. Though helpful, there are open research questions about how best to compare multiple models directly. However, these are not fully addressed in the current design.

7 Conclusion

In this paper, we described a technique for interactive multi-model steering and inspection. The technique emphasizes the importance of moving beyond existing, single-model steering techniques where the initial model choice greatly impacts the quality of the results. Instead, our technique steers and samples from multiple model types, learning algorithms, and hyperparameters. Our visual analytics prototype, BEAMES, allows people to specify interest in models, data instances, and attributes to iteratively build, combine, and select models to perform prediction tasks. We describe our technique and demonstrate the use of BEAMES through a use case scenario, highlighting the model steering and inspection process resulting from user interactions. Finally, we discuss broader challenges in the area of multi-model steering and illuminate valuables areas for future work.

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] H. Afrabandpey, T. Peltola, and S. Kaski. Interactive prior elicitation of feature similarities for small sample size prediction. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, UMAP '17, pages 265–269, New York, NY, USA, 2017. ACM.

[2] 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.

[3] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 337–346, New York, NY, USA, 2015. ACM.

[4] 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*, pages 13–20, 2013.

[5] F. Bouali, A. Guettala, and G. Venturini. Vizassist: An interactive user assistant for visual data mining. *Vis. Comput.*, 32(11):1447–1463, Nov. 2016.

[6] 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)*, pages 163–172, Oct 2014.

[7] L. Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, Aug. 1996.

[8] E. T. Brown, J. Liu, C. E. Brodley, and R. Chang. Dis-function: Learning distance functions interactively. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 83–92, Oct 2012.

[9] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[10] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 18–, New York, NY, USA, 2004.

[11] J. C. Chang, S. Amershi, and E. Kamar. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 2334–2346, New York, NY, USA, 2017. ACM.

[12] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836, 1979.

[13] M. W. Craven. *Extracting Comprehensible Models from Trained Neural Networks*. PhD thesis, 1996. AAI9700774.

[14] P. Daee, T. Peltola, A. Vehtari, and S. Kaski. User modelling for avoiding overfitting in interactive knowledge elicitation for prediction. In *23rd International Conference on Intelligent User Interfaces*, IUI '18, pages 305–310, New York, NY, USA, 2018. ACM.

[15] S. Datta and E. Adar. Communitydiff: Visualizing community clustering algorithms. *ACM Trans. Knowl. Discov. Data*, 12(1):11:1–11:34, Jan. 2018.

[16] 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.

[17] A. Endert, P. Fiaux, and C. North. Semantic interaction for sensemaking: Inferring analytical reasoning for model steering. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2879–2888, Dec 2012.

[18] 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, pages 473–482, New York, NY, USA, 2012. ACM.

[19] 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*, pages 121–130, 2011.

[20] 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*, pages 2962–2970, 2015.

[21] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, pages 148–156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[22] M. Gleicher. A framework for considering comprehensibility in modeling. *Big Data*, 4(2), Jun 2016. ahead of print.

[23] K. inc. Kaggle Housing prices Data. `https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data/`, 2018. [Online; accessed 15-July-2018].

[24] G. Jawaheer, P. Weller, and P. Kostkova. Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. *ACM Trans. Interact. Intell. Syst.*, 4(2):8:1–8:26, June 2014.

[25] 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*, volume 28, pages 767–774. Wiley Online Library, 2009.

[26] M. P. P. Jr. Combining classifiers: From the creation of ensembles to the decision fusion. In *2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials*, pages 1–10, Aug 2011.

[27] A. Jurek, Y. Bi, S. Wu, and C. Nugent. A survey of commonly used ensemble-based classification techniques. *Knowledge Engineering Review*, 29(5):551581, 2013.

[28] 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.

[29] H. Kim, J. Choo, H. Park, and A. Endert. Interaxis: Steering scatterplot axes via observation-level interaction. *IEEE Transactions on Visualization & Computer Graphics*, 22(1):131–140, Jan. 2016.

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

[31] B. Kondo and C. Collins. DimpVis: Exploring Time-varying Information Visualizations by Direct Manipulation. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2003–2012, Dec. 2014.

[32] 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.

[33] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, New York, NY, USA, 2004.

[34] B. C. Kwon, B. Eysenbach, J. Verma, K. Ng, C. D. Filippi, W. F. Stewart, and A. Perer. Clustervision: Visual supervision of unsupervised clustering. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):142–151, Jan 2018.

[35] 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 Transactions on Visualization & Computer Graphics*, 23(1):221–230, Jan. 2017.

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

[37] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit. Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE Transactions on Visualization & Computer Graphics*, 20(12):1643–1652, Dec 2014.

[38] S. Pajer, M. Streit, T. Torsney-Weir, F. Spechtenhauser, T. Mller, and H. Piringer. Weightlifter: Visual weight space exploration for multi-criteria decision making. *IEEE Transactions on Visualization & Computer Graphics*, 23(1):611–620, Jan 2017.

[39] S. Paparizos, J. M. Patel, and H. V. Jagadish. SIGOPT: using schema to optimize XML query processing. In *ICDE*, pages 1456–1460. IEEE Computer Society, 2007.

[40] K. Patel, S. Drucker, J. Fogarty, A. Kapoor, and D. Tan. Using multiple models to understand data. pages 1723–1728. AAAI Press, July 2011.

[41] N. Pezzotti, B. P. F. Lelieveldt, L. van der Maaten, T. Hllt, E. Eisemann, and A. Vilanova. Approximated and user steerable tsne for progressive visual analytics. *CoRR*, abs/1512.01655, 2015.

[42] H. Piringer, W. Berger, and J. Krasser. Hypermoval: Interactive visual validation of regression models for real-time simulation. *Comput. Graph. Forum*, 29:983–992, 2010.

[43] K. Potter, A. Wilson, P. T. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. R. Johnson. Ensemble-vis: A framework for the statistical visualization of ensemble data. In *2009 IEEE International Conference on Data Mining Workshops*, pages 233–240, Dec 2009.

[44] D. Ren, S. Amershi, B. Lee, J. Suh, and J. Williams. Squares: Supporting interactive performance analysis for multiclass classifiers. IEEE, August 2016.

[45] 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.

[46] M. A. Rene Cutura1, Stefan Holzer and M. Sedlmair.

[47] 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.

[48] B. Saket, H. Kim, E. T. Brown, and A. Endert. Visualization by demonstration: An interaction paradigm for visual data exploration. *IEEE Transactions on Visualization & Computer Graphics*, 23(1):331–340, Jan 2017.

[49] B. Schneider, D. Jäckle, F. Stoffel, A. Diehl, J. Fuchs, and D. A. Keim. Visual integration of data and model space in ensemble learning. *CoRR*, abs/1710.07322, 2017.

[50] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Möller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization & Computer Graphics*, PP, 2014.

[51] 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, pages 82–91, New York, NY, USA, 2007. ACM.

[52] 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.

[53] J. Talbot, B. Lee, A. Kapoor, and D. S. Tan. Ensemblematrix: Interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1283–1292, New York, NY, USA, 2009. ACM.

[54] 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*, pages 847–855. ACM, 2013.

[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*, pages 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.

[57] 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.

[58] 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, pages 14:1–14:6, New York, NY, USA, 2017. ACM.

[59] D. H. Wolpert. Original contribution: Stacked generalization. *Neural Netw.*, 5(2):241–259, Feb. 1992.

[60] Q. Yang, J. Suh, N.-C. Chen, and G. Ramos. Grounding interactive machine learning tool design in how non-experts actually build models. In *Proceedings of the 2018 Designing Interactive Systems Conference*, DIS '18, pages 573–584, New York, NY, USA, 2018. ACM.