Eurographics Conference on Visualization (EuroVis) 2020
M. Gleicher, T. Landesberger von Antburg, and I. Viola
(Guest Editors)

Volume 39 (2020), Number 3

Boxer: Interactive Comparison of Classifier Results

Michael Gleicher, Aditya Barve, Xinyi Yu, and Florian Heimerl

University of Wisconsin – Madison

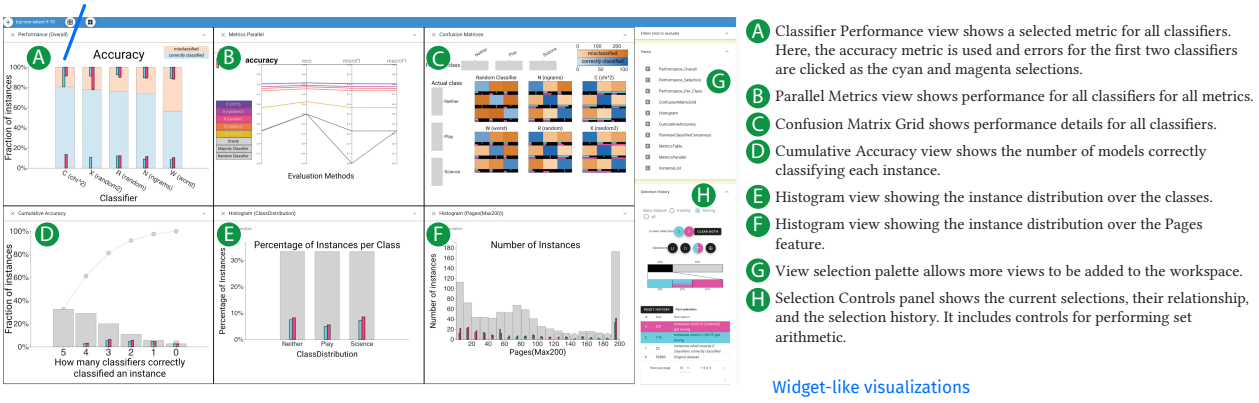

Figure 1: *Boxer examining the results of 5 classifiers in the Feature Selection use case of Section 5.2.*

Abstract

Machine learning practitioners often compare the results of different classifiers to help select, diagnose and tune models. We present Boxer, *a system to enable such comparison. Our system facilitates interactive exploration of the experimental results obtained by applying multiple classifiers to a common set of model inputs. The approach focuses on allowing the user to identify interesting subsets of training and testing instances and comparing performance of the classifiers on these subsets. The system couples standard visual designs with set algebra interactions and comparative elements. This allows the user to compose and coordinate views to specify subsets and assess classifier performance on them. The flexibility of these compositions allow the user to address a wide range of scenarios in developing and assessing classifiers. We demonstrate Boxer in use cases including model selection, tuning, fairness assessment, and data quality diagnosis.*

Focused on data subsets/slices as a way to be more granular than performance metrics

CCS Concepts

• ***Human-centered computing*** → *Visualization; Visual analytics; Information visualization;*

Focused on model comparison (model comparison via data subsets)

1. Introduction

Machine learning practitioners often perform experiments that compare classification results. Users gather the results of different classifiers or data perturbations on a collection of testing examples. Results are stored and analyzed for tasks such as model selection, hyper-parameter tuning, data quality assessment, fairness testing, and gaining insight about the underlying data. Classifier comparison experiments are typically evaluated by summary statistics of model performance, such as accuracy, F1, and related metrics. These aggregate measures provide for a quick summary, but not detailed examination. Examining performance on different subsets of data can provide insights into the models (e.g., to understand performance for future improvement), the data (e.g., to understand data quality issues) or the underlying phenomena (e.g., to identify potential causal relationships). Relying on aggregated data can miss important aspects of classifier performance. For closer examination, practitioners rely on scripting in their standard workflows. The lack of specific tooling makes the process laborious and comparisons challenging, limiting how often experiments are examined in detail.

This paper presents *Boxer* (Figure 1), a system for the detailed examination of classifier comparison experiments. Our approach allows a user to explore a collection of classifier results to identify interesting subsets of the data and compare performance across them. Our system enhances standard views with interactions for selection and comparison. The design provides a uniform mechanism for identifying subsets, choosing appropriate metrics, and assessing performance over different parts of the data. Users combine views and build selections to pose comparisons for visual assessment.

Our work applies to the results of classifier experiments: the sys-

tem operates on pairs of classifier inputs (testing and training data) and outputs (the predictions made). Boxer treats classifiers as black boxes: it does not consider the internals of how the classifiers work, allowing it to be applied broadly and compliment existing, specialized tools. To show the potentially large data, the system provides a set of summary views. These views are variants of familiar displays, such as bar charts and confusion matrices.

Boxer is built around the unifying abstraction of a *box*. Boxes are a "container" for a subset of data elements, and a mark [Ber10, Wil05] that shows information about this subset. They thus couple a data abstraction with a graphical abstraction. A visual element, such as the bar of a bar chart or the square of a confusion matrix (Figure 2), connects input (select subset), display (show subset size), and comparison (relate subset to others). Users create *selections* of data elements of interest by selecting such boxes. The selections are displayed relative to all boxes across the interface: users can see the overlap between the section and other boxes. Boxer supports dual active selections, allowing for quick comparison and building more complex selections by combining simpler ones using binary set operators (e.g., intersection, union). For example, in Figure 1, the user has selected the top parts of the leftmost bars in the Accuracy view (A) as the first (cyan) and second (magenta) selections. These boxes represent the instances classified incorrectly by the two classifiers. Cyan and magenta glyphs throughout the interface show the overlap of these selections with other boxes. These selections can be combined, for example by intersecting their contents to identify instances classified incorrectly by both classifiers.

In summary, the three main innovations of Boxer are: (1) an approach to classifier comparison that combines subset identification, metric selection, and comparative visualization to enable detailed comparison; (2) an architecture of multiple selections and set algebra that allows users to flexibly link views and specify data subsets of interest; and (3) interaction techniques and visual designs that make the approach practical. These ideas should be applicable in other systems for interactive comparison.

2. Related Work

Interactive tools for understanding machine learning models are motivated by many reasons, see Gleicher [Gle16] or Lipton [Lip16] for surveys. The range of needs is addressed by approaches that can be roughly categorized into four groups:

1. Transparent models use learning representations designed for easy examination, such as generalized additive models [LCGH13, CLG*15], rule-based learning [WR15, OW16], RETAIN [CBS*16], approximating decision trees [MBVV07, CS68, Cra96, CS96, MQB19], and sparse linear models [Gle13].
2. Per-instance explanations provide methods that explain individual decisions or local groups, examples include expert system tracing [Wei80, Wal84], variable sensitivities [TKDB17], influential feature identification [SDV*16, AHM*17], locally linear models [RSG16], and instance examination [KDS*17].
3. Internal inspection approaches provide tools to examine specific types of complex models such as neural nets [SGPR18, LSL*17, MCZ*17]. Variants of these approaches exploit knowledge of model structure, such as the Treepod [MLMP18] system that helps users understand trade-offs in decision tree classifiers.
4. Black box methods which do not consider the internals of the model, but instead rely on observations of their input/output pairs. *Our work falls into this last category, providing a general approach that works with a range of model types.*

Spinner et al. [SSSE20] develop a pipeline for model analysis and comparison that organizes such methods into three categories that focus on model input, model output, and model internals. In the latter category, a range of methods have been proposed to "open up" those black boxes and provide a view of the inner workings of ML models. Examples include representations of prototypical data instances [MV16, YCN*15, AB16], structural overviews of complex models [LSC*18, WSW*17], and explaining the influence of model structure on its output [HPRP20, RFFT17, ZF14, AR15]. DeepCompare [MMD*19] allows instance-level comparisons of deep learning models and connecting decisions back to structural properties of the models. *In contrast, our work is strictly black box and focuses on using the results of existing experiments on models, and making comparisons between multiple models.*

Black box methods are often designed for very specific goals. Ma et al. [MXLM20], for example, support the identification of feature combinations that elicit a specific response from a model. Ye et al. [YXX*19] enable users to assess and increase the quality of training data labels. Another important goal is the analysis of fairness of predictions. Friedler et al. [FSV*19] review classifier fairness and provide measures. Cabrera et al. [CEH*19] present a method for analyzing fairness by generating subsets of the data with different prediction performance. In contrast, Ahn and Lin [AL20] focus on identifying instance-level bias. *In contrast, our approach supports a broad range of tasks with flexible mechanisms to analyze different aspects of data and classifier performance.*

Some prior black box approaches support a range of tasks. ModelTracker [ACD*15] is an interface designed to provide an overview of model performance and detailed inspection of instances and their features through interaction. *In contrast, we provide a range of views that are linked through subsets of the classification data and allow in-depth comparisons between models.* The What-If-Tool [WPB*20] enables users to compose a range of visualizations, including bar chars and confusion matrices on subsets of their data by slicing sets based on feature values. *In contrast, we enable the creation of more complex subsets based on set algebra, and enable comparisons between models based on these sets.* Manifold [ZWM*19] provides specialized variants of standard displays to show correlations between model decisions and their relation to the underlying data. *In contrast, we provide more flexible composition of views with comparative elements and selection construction.*

Black box methods are also used to test the stability and robustness of a trained model [Bre01]. Such sensitivity analysis methods perturb the values of input features. They can identify relationships between outputs and features [HPB*14, ŠK14], even in complex models [SGK17, OJ02, STY17]. Lee et al. [LSC19] provide a systematic approach to test for the contributions of single features to model errors, and for potential interactions between features. A conceptually similar approach is generating prototypical inputs for a model [SVZ13]. *Our approach is designed to examine the results of such experiments.*

There is a long standing effort in ML to develop metrics

for assessing model performance [Pow11, WFH11]. Some metrics address specific tasks such as interpreting the performance changes [SVC*10]. Similarly, visualization approaches extend basic chart types (e.g., ROC curves and confusion matrices) to present more information and serve as interactive tools for achieving specific user tasks. For example, tools exist for characterizing errors in regression models [MP13], examining classification results within their original context [SAMG14], adjusting weights from confusion matrices [KLTH10, TLKT09], and showing probability distributions in multi-class classifiers [AHH*14]. In particular, Squares [RAL*17] enables in-depth comparison of classification models based on visual summaries of instance-level predictions and classifier probabilities. However, the approach does not support subsetting of the dataset and comparisons between subsets. *In contrast, our work focuses on comparing models by identifying the subsets of instances that make up these metrics.*

Visual comparison approaches have been developed for many model types. For example, approaches exist for specialized types of models, such as sequence models [MXC*20, SGB*19], or for specific data types, such as set data [AMA*14]. Other approaches focus on clustering [KEV*17, CD19], topic models [AG16], word vector embeddings [HG18], and climate models [DWOB20]. *In contrast, our work focuses on making comparisons between discrete choice classifiers from existing classifier experiment data.*

3. Comparison of Classification Models

We consider comparing the results of classifier experiments. A *classifier* is a machine learning model that predicts a *label* from a data *instance*. Supervised machine learning builds classifiers using a *training set* of instances for which the labels are known. Classifiers may be evaluated by assessing their predictions on this training set (known as *resubstitution*), but are more commonly evaluated on a *testing* or *hold-out* set of instances. A *classifier experiment* assesses the performance of one or more classifiers over a set of instances. An experiment may involve multiple classifiers, comparing performance on a common testing set, or involve the same classifier with multiple variants of the testing set.

The results of a classifier experiment are a set of data instances, a correct label for each instance, and, for each classifier, a prediction. Boxer uses this data as input. The data instances themselves are useful, as many tasks involve building connections back to the underlying data. However, the measure in most classifier experiments is correctness: whether the predictions match the correct labels. These results are summarized by counting the different outcomes across a set of answers. A *confusion matrix* counts all possible outcomes (if there are l labels, there are $l \times l$ possible outcomes to be counted). Metrics of classifier performance reduce this matrix to a single number. For example, *accuracy* counts the number of correct predictions (the prediction matches the correct label), and divides by the total number of instances. Other metrics, such as F1, precision, recall, error rate, and mathews coefficient, summarize the counts of confusion matrices in different ways. See Chapter 5 of Witten, et al. [WFH11] or Powers [Pow11] for comprehensive discussions. Even more metrics are possible if additional information, such as confidence, is available for the predictions. Selecting an appropriate metric is an important part of assessing classifiers.

3.1. Tasks of Classifier Comparison

Many tasks in classifier development and assessment involve comparison of classifier results. Model selection, tuning, and fairness assessment rely on examination of testing results between classifiers. Even the selection of an appropriate metric can be informed empirically. Other tasks, such as data quality assessment, feature engineering, and gaining insights from data can make use of a collection of classifiers and their results.

However, classifier comparison typically considers summary metrics over the entire testing set. This hides information about which instances different classifiers get right or wrong. This information may be useful across a range of tasks. For example:

- In *model selection,* having information beyond summary metrics can determine performance in more important and relevant cases. This can help adapt test metrics to predict real world performance.
- In *model tuning,* understanding where a model works (or doesn't) can help choose strategies for improving performance and confirm that interventions work as expected.
- In *fairness assessment,* different subgroups of instances can be compared to identify ones being treated unfairly.
- In *data quality assessment,* identifying interesting instances can point to problems or opportunities in the underlying data. For example, view can show that errors as associated with missing data.
- In *studying the underlying data,* identifying specific instances or differences between groups can help test theories or identify underlying mechanisms behind the data.

We will provide examples of such scenarios in Section 5.

The range of tasks that can use subset selection and assessment is broad and diverse. However, in working with practitioners performing these tasks, a core pattern of elemental operations emerge. Users must (1) identify an appropriate subset of the instances; (2) select an appropriate metric; (3) compare performance across classifiers; and (4) relate these results in both the broader context as well as specific details. This process is often exploratory and iterative: a user examines a metric across a number of subsets, making comparisons that suggest different combinations. The typical practice of using scripting to query data to create subsets and perform evaluations does not support rapid exploration or visualization that enhances comparison or helps contextualize results.

Rather than designing specific support for the broad range of tasks, our strategy is to provide flexible support for elemental operations common across tasks, and provide mechanisms to combine these operations to support workflows that address tasks.

3.2. Boxes and Selections

Many classifier comparison explorations involve two basic operations: (1) defining subsets of instances, and (2) comparing performance and specific predictions between these subsets. By composing these two elements, we can construct many more complex tasks. For example, by comparing the subset of instances a classifier gets wrong with the subset formed by histogram bins (e.g., of

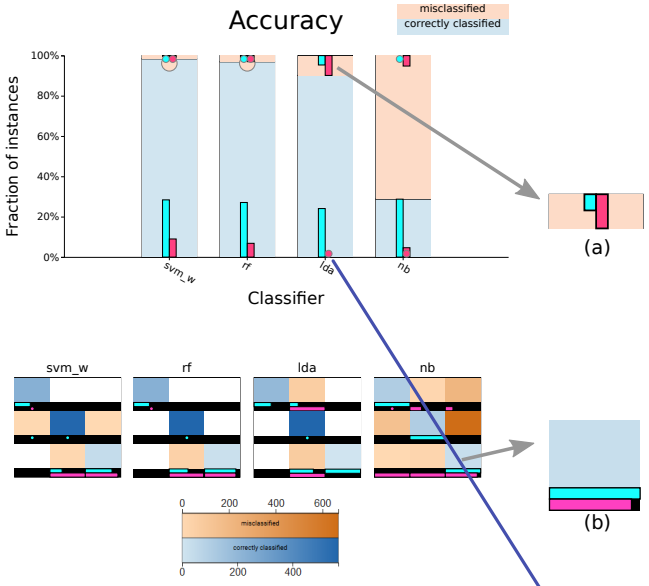

Figure 2: *Visualizations composed of* boxes *and* selections*: Each bar of a bar chart (a) and square of the confusion matrix (b) is a box corresponding to a subset of the data. All boxes show their overlaps with the current selections with color coded stripes. Dots in bars indicate small non-zero values.*

In a bar chart, for example, use a different shape, such as a circle, for very small values (whose bar would be tough to see). If there is no shape, it is clear that the value is zero.

a feature), we can see where in the dataset an error occurs. Our design focuses on providing fluent and flexible support for these basic operations.

We support subsets through the abstractions of *boxes* and *selections*. A *box* connects a data abstraction to a visual abstraction. A box represents a specific subset of the instances, which can be defined as a query over the entire collection. A box also connects to a visual element that summarizes the subset. For example, a bar of a bar chart represents a subset of instances (the query is the x range of the bar) *and* a visual element (the rectangle encoding the count of the subset with height). A *selection* is a subset of instances of interest to the user. At any given time, there may be an *active* selection that is being explicitly highlighted.

Boxes and active selections connect: boxes serve to create selections (e.g., clicking on a box sets the active selection), and the active selection can be presented visually in each box to enable comparison by showing the intersection between the box and the selection. This is possible because each box represents a subset of the data. The queries serve as textual descriptions that describe selections, for example in history views. Our system supports two active selections. Both selections are shown in all boxes (Figure 2). The left and right mouse button are used to make corresponding selections. Consistent coloring is used throughout the interface: cyan for the first active selection, magenta for the second. We use vivid colors that contrast other interface elements even at small sizes [Sza18]. Multiple selections extend prior abstractions of aggregation (e.g., [SDW09, EF10]) to better support composition and comparison.

Dual active selections are a key feature of Boxer. A single active selection does not allow for comparisons between selections, and

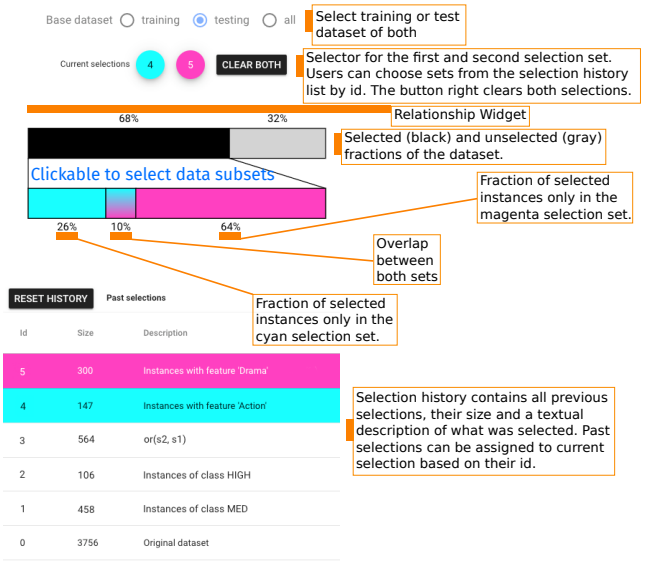

Figure 3: *The* Selection Control *view provides information about the active selections and controls to adjust them. The view provides a textual description of the active selections, as well as a visual indicator that relates these subsets to the overall data set and to each other. The lower row of the indicator shows the relation between the two active selections, including their intersection and differences. Clicking on the indicator selects that subset, allowing for easy set operations (e.g., clicking on the middle area to select the intersection). The Selection Control view provides a history list of previous selections, which can be clicked to recall one.*

Preclude: prevent from happening

does not allow for an interface for selection composition. More than two active selections may enable richer comparisons, but comes at the expense of visual clutter, the need to reserve more colors (precluding their use elsewhere in the system), and a need to find new methods for showing and interacting with the sections. Dual selections are sufficient for creating complex subsets by composing binary set operations (e.g., intersection, union, subtraction).

Information about the active selections are shown in the Selection Control view (Figure 3). This view provides a textual description of current and previous selections, a history of previous selections allowing them to be recalled by clicking, and a widget that shows the relationship between the two active selections. The Selection Control view allows for set arithmetic operations to be performed between the two selections using the relationship widget. For example clicking on the overlap of the two selections creates a new selection that is their intersection.

The box abstraction helps create a uniform and flexible mechanism for expressing set comparison as well as a consistent user experience. Boxes in our system support four features. First, they have an associated *query* that finds the instances in their subset and allows for textual description. Second, they have some *visual representation* that displays the size of the set. Third, the visual elements of a box are *clickable,* serving as input to identify that the subset associated with the box is of interest to the user. Fourth, each box displays a *comparison* with the active selections. Our system sup-

ports two active selections, so each box must show three quantities: the number of instances in the box's associated subset, the size of the intersection of the box's set with the first selection, and the size of the intersection of the box's set with the second selection. Examples are shown in Figure 2.

Each mark is a box

Each rectangular region of a bar chart, stacked bar chart, or confusion matrix is a box. Other visualizations, including pie charts, hexbin plots, and treemaps, could similarly provide a set of regions, each corresponding to a box. The four features of boxes give an interaction mechanism and visual grammar for displaying comparisons. All visualizations built from boxes serve as mechanisms to specify selections, and to make comparisons between the set defined by the box and other sets of interest to the user.

Usage

Users build selections by selecting boxes and combining them with binary set operations to make more complex selections. Users make detailed comparisons by seeing how the selections compare to each other or against boxes across different views.

4. Boxer Views

Boxer provides users with a number of view types that can be combined in the workspace as needed. The box mechanism allows for specifying sets of interest and performing comparisons in these familiar views, allowing users to couple simple displays to address complex tasks. Our design explores the use of a few basic views and a general composition mechanism, rather than designing more specialized views. Other view types could be added in the future. However, introducing new views has a cost: users must be able know when to apply them. Boxer had other views that have not proven useful in practice, and were removed.

4.1. Bar Chart-based Views

Many of Boxer's main views use bar charts that divide the instances in different ways. Bars correspond to boxes: they can be used to select their contents as well as to display their overlap with the two active selections, as shown in Figure 2. Boxer's bar charts illustrate non-zero values with circles (e.g., Figure 4.2, 4.4, 8) which are hidden if the bar becomes large enough. This allows small subsets (such as rare instances) to be seen and selected as the data grows. Charts allow for normalization to enable comparisons between sub-bars (Figure 8).

The *Classifier Performance* view (Figure 1a, 4.1, 5 left) shows the performance of each classifier in a stacked bar. A variety of metrics can be chosen. For metrics that are ratios of subset counts (e.g., accuracy, precision, and recall), the bars are stacks of boxes. This view provides a simple overview of classifier performance, and an easy way to select sets of instances (e.g., what classifier predicts correctly). The view allows sorting by value to facilitate identification and comparison of the best or worst classifiers.

The *Histogram* (Figure 1e, 1f, 4.2-5) view shows the distribution of the data across a feature. This includes the data features as well as the actual and predicted classes. The user can place multiple histograms to show different distributions. Continuous features are bucketed, and discrete features can be sorted in various ways (e.g., sort by quantity to emphasize the largest categories). Histograms provide an important mechanism for selection as well as comparative display. Combining bars from multiple histograms using set operations can specify complex selections.

The *Cumulative Accuracy* view (Figure 1d, 4.2), shows how many classifiers correctly labeled each data instance. For example, the viewer can identify items that no classifiers predicted correctly or that all classifiers predicted correctly. This view may be used to select challenging instances, or to see if a selected set contains easy items. The Cumulative Accuracy view includes a pareto line allowing the user to quickly assess and select the cumulative sum.

The *Selection Performance* view (Figure 6g, h, i, and 5 right) shows the performance of each classifier across both selections. The user can select a variety of different metrics (accuracy, F1, etc.). Because the bars are associated with selections (and colored accordingly), they do not serve as boxes. In contrast, the *Per-Class Performance* view provides boxes while showing the performance of each classifier for each of the actual classes of the instances. The user selects a metric (accuracy, F1, etc.) for this faceted bar chart.

4.2. Matrix-based Views

Matrix-based views arrange boxes in a fixed grid. Because the size is fixed, color encodes for the size of the subset the box corresponds to. The color encoding allows for rough comparison between squares: detailed values can be revealed by hovering. Small bars, color-coded to match the selection scheme, show the overlap between the active selections and the box's subset – a full bar means that all of the box's instances are in the selection. Matrix cells can be clicked for selection.

The *Confusion Matrix Grid* view (Figure 1c) provides the standard view of classification results for each classifier. It shows each classifier's performance, broken down by label. This allows us to compare classifiers based on their prediction profile per class.

The *Pairwise Consensus* view shows the agreement and disagreement between each pair of classifiers as a matrix. It conveys the number of instances for which two classifiers predict the same label. The matrix is split on the diagonal to distinguish agreement on correct vs. incorrect instances. This view is useful in identifying correlations between classifiers, for example to assess ensembling.

4.3. Other Views

Boxer provides views designed to help with metric selection. The *Standard Metrics* view provides a simple table of many metrics across all classifiers. The *Parallel Metrics* view (Figure 1b) presents the same information in a parallel coordinates chart. Similar to [DWOB20], parallel coordinates help identify correlation between metrics. The view uses an ordered coloring based on a selected metric to aid in order comparison.

The *Instance List* view shows a tabular display of the instances in the active selections. Instances are color-coded to indicate which selections they are part of. Users can select instances, allowing for fine-grained, instance-level modification of selections.

The *Selection Controls* panel was described in Section 3.2 and Figure 3. In addition to selection display and interaction, it allows

users to switch between training set, selection set, or both. The Boxer interface allows viewing all data instances, or limiting the views to either training or testing data.

5. Use Cases

This section provides some example scenarios where Boxer can be used to address tasks in classifier development. These examples are chosen to highlight how Boxer's key ideas can be applied in common machine learning settings. The examples come from student data scientists working with standard data sets, as well as from our collaboration with literature scholars to use statistical techniques to help analyze historical text corpora.

All examples were completed using the Boxer system. Boxer is built as a web application, written in TypeScript and using the Vue.js application framework and D3.js for drawing visualizations. The system loads all data, both classification results as well as feature and meta-data for all data instances into memory at startup and performs all computations within the browser. Boxer is efficient enough to handle data sets with tens of thousands of instances. We discuss scalability issues in Section 6.1.

5.1. Model Selection and Tuning: Movies

We consider a simple use case that shows how Boxer's views can be combined to diagnose classifier problems. We use boxer in developing a classifier to predict movie ratings from IMDb. Our data consists of 5,044 movies with 27 features. 25% are sequestered for final assessment. Classifiers predict a movie's rating (low, medium, high). A stratified sampling of 200 movies per class is held out from the 75% remaining. A variety of classifiers were constructed, none with acceptable accuracy.

Using boxer, we examine the results (Figure 4). The Classifier Performance view confirms the poor performance. The Cumulative Accuracy shows large numbers of instances that are easy and hard (all or no classifiers are correct). We select these easy and hard subsets. In a Histogram view, we can see the hard elements are in the *high* class. Examination of the training set shows that there are few examples of this class. We build a new classifier (offline) that accounts for this skew. Using boxer, we can confirm that this has superior performance, although its errors are still biased. While conventional tools can show skew, the example shows how Boxer's flexible mechanisms allow performance effects to be connected to data issues.

5.2. Model Selection and Data Discovery: Literary Features

This use case considers a corpus of 59,989 documents from a historical literary collection: Text Creation Partnership (TCP) transcriptions of the Early English Books Online (EEBO). Of these documents, 1,065 have been identified as *plays*, 1,974 as *science* documents, and most are *neither*. The data counts the 500 most common English words in each document. While all documents have been classified by experts, we construct classifiers using the data to support theories that different types of documents use words in different ways [WH10, Gle13]. Specifically, we are interested if a small set of words can identify document classes. Boxer allows us to compare the performances of classifiers built from different sets of words to confirm the impact of word choice on performance.

We create decision tree classifiers using a variety of univariate feature selection strategies. Each selects 10 words to count for features. The feature selection methods were: most relevant by a CHI-squared univariate feature selector (C), most common features (N), randomly chosen features (R), and, as a baseline, the features deemed worst (out of the 500 candidate words) by the CHI-squared test (W). A testing set of 200 documents per class was used.

We compare the results in Boxer (Figure 1). The Parallel Metrics view shows a consistent ordering of the classifiers across all metrics: C is slightly better than R and N, which are much better than W. Using the Classifier Performance view to see the accuracy details, we can select the mistakes made by the top classifiers (cyan for C's mistakes, and magenta for R's mistakes). We see that the errors are relatively evenly distributed among the classes in a Histogram view of the class distribution. This is surprising given the skewed training distribution. We also see in the Performance view that different classifiers make different errors (e.g., only half of C's errors are made by N). Overall, the Cumulative Accuracy view shows that there are very few instances that all classifiers were wrong on (2.6%), and a Histogram view of document lengths lets us see that performance is relatively consistent over the range of document lengths.

5.3. Feature Sensitivity Testing: Plays

We use Boxer to provide more insight into the results of a variable sensitivity experiment. The data set is a collection of 554 plays written in the Early Modern Period (1470-1660). Five linguistic features (selected form [IK11]) are used. We classify plays with one of four genres (Comedy, History, Tragedy and Tragicomedy). Ground truth is known; the goal is to use the classifier to determine relationships between the linguistic features and genre [WH10, Gle13]. Our experiment uses a Support Vector Machine (SVM) classifier trained with class weights to counteract a skewed training distribution. A stratified sample of 20% was removed as the test set. The training set has very few of the underrepresented classes.

After training, a feature sensitivity experiment identifies which features contribute to the classifier's performance. We create a variant of the data set for each feature. In each data variant, small perturbations (positive and negative) are added to its corresponding feature's value for all entries in the data set. The resulting data sets have twice as many items as the original (one for positive additions to the feature, one for negative ones). Each data variant is run through the classifier. Here, we consider only the testing set. Such experiments are preferred to simply examining model coefficients because they test the effects of the variables near the actual data.

The standard approach to analyzing such an experiment is to compute summary statistics over each feature's data variant and compare these to the baseline. More advanced approaches [LSC19] can check for the statistical significance of these differences. The experiment results show that for feature *Negavity* (N), the classifier performs much worse than the baseline using standard metrics (accuracy and Mathews correlation). Two features *PersonProperty*

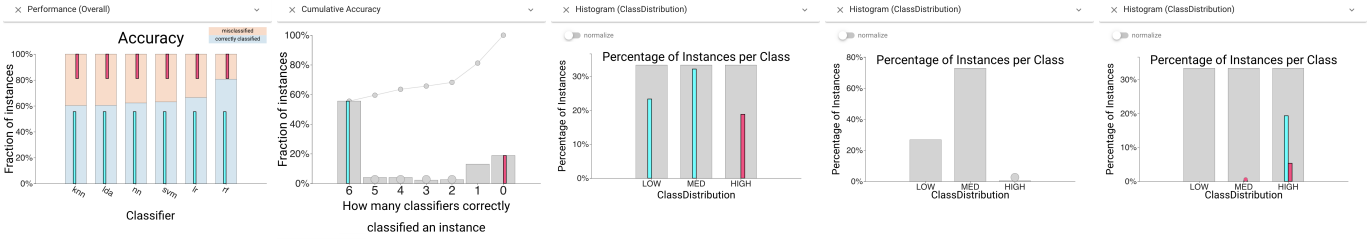

Figure 4: *Movie model selection use case (Section 5.1). From left to right: (1) The Classifier Performance view shows low performance for all classifiers; (2) We then select easy (cyan) and hard (magenta) instances in the Cumulative Accuracy view; (3) A Histogram view of the test set shows us that all hard (magenta) instances are in the* high *class; (4) A Histogram view of the training set shows class skew; (5) A Histogram view shows that the errors of the new classifier (magenta) are still biased, but less than the best prior classifier (cyan).*

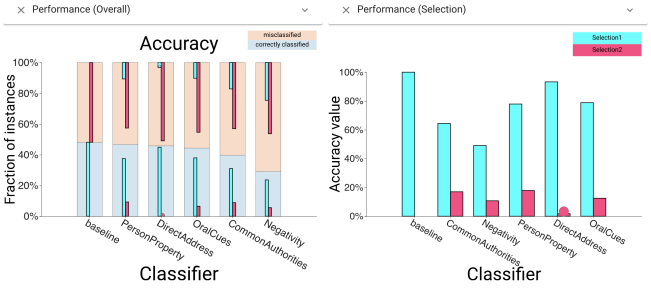

Figure 5: *Sensitivity testing use case (Section 5.3). The* Classifier Performance *view (left) shows that (PP and DA) perform similar to the baseline. Selecting baseline correct (cyan) and incorrect (magenta) shows considerable overlap with DA, but not PP. This suggests the model is sensitive to PP, despite similar aggregate performance to the baseline. The* Selection Performance *view (right) shows this as well: PP achieves 20% accuracy on the baseline's errors.*

(PP) and *DirectAddress* (DA) achieve similar performance to the baseline. Standard procedure would conclude that the classifier is sensitive to N but not PP and DA. However, closer examination in Boxer reveals otherwise. The Classifier Performance view (Figure 5) shows that PP and DA have similar accuracy to the baseline. Selecting the correct and incorrect subsets for the baseline allows us to compare with the perturbed results. For DA, the overlaps are substantial, for PP there is less overlap. While PP gets the same number of instances correct, it is correct on different ones, suggesting the model is sensitive to this feature.

5.4. Model Selection: Mushroom Imputation

We consider adapting a classifier to identify poisonous mushrooms. The initial training of the baseline classifier uses the *color* feature, but we would like to build a new classifier that does not use this feature. We consider two approaches to imputing the missing feature: using the mode of the data and training a decision tree to classify the color based on other features. In addition to the baseline classifier, we build two new models: *mode* and *smart* that use the imputed versions of the color feature. Note that mode is effectively not using the color feature, as it is constant across its data set.

We assess these strategies with a testing set of 2,000 (of 8,124) randomly selected instances. The Parallel Metrics view shows that the imputed models perform worse than baseline on all metrics. The smart model performs better than mode on all metrics except recall, which is likely to be important (we don't want to eat a poisonous mushroom). To understand these differences, we can select the instances where the baseline and smart classifiers are incorrect and see that the latter is almost a proper subset of the former.

We wish to understand if the lower performance of the smart classifier can be attributed to imputation mistakes. We select the instances where the baseline is correct and where smart is wrong, and intersect them. Examining these sets in a Histogram view of the color feature, we see that most of the errors are white mushrooms. Looking at the imputed feature over this set, we see that the smart imputer never labels these as white. These mistakes of the imputer likely cause the misclassifications.

5.5. Fairness Assessment: Recidivism

We consider a standard test case for fair learning: the Broward County recidivism dataset, popularized by ProPublica [ALMK16]. This data set was initially used to show unfairness of a commercial system, but has emerged as a benchmark for machine learning fairness [FSV*19]. The task is to predict whether a person will commit a crime within two years (two year recidivism). The data set includes ground truth. Classifiers built for this problem are often *unfair* in that they skew errors towards racial and gender bias. Specialized tools, such as FairVis [CEH*19], are designed for assessing classifier fairness. In this use case, we show how Boxer's flexible mechanisms can be used for similar purposes.

The data set contains 6,172 instances (chosen by the criteria of [FSV*19]) and 14 numeric features (created by one-hot encoding the categorical features in the initial seven feature data set). 20% are held for testing. We consider three classifiers trained on the data, a *baseline* random forest, and two hand-tuned variants (*C3* and *Pos*). In the Parallel Metrics view, we can see that C3 achieves higher scores for all metrics. We question whether it achieves these improvements in a fair manner.

Analysis of this case is shown in Figure 6. We use a Histogram view of the *race* feature to select Caucasian and African-American instances. Various views in boxer clearly show the unfairness.

Parallel Metrics view A shows the C3 classifier has better performance by all metrics.

A histogram of race F selects Caucasian(cyan) and African-American (magenta) instances.

The Overall Performance view B shows C3's overall higher precision, but a lack of overlap with cyan.

The Confusion Matrix Grid view C shows many false positives for African-Americans and many false neg. for Caucasians for C3.

Histograms show the distribution of selected sets across the actual D and the C3-predicted class E.

The Performance Selection views in the third line compare accuracy G, precision H, and recall I for C3 on the subsets.

Figure 6: *Boxer in the recidivism use case (Section 5.5).*

While a Selection Performance view shows similar accuracies for the selections, the precision and recall are very different. C3 has high precision but low recall for Caucasians, and high recall but low precision for African-Americans. That is, its errors are biased to predict *no* for Caucasians and *yes* for African-Americans. While other classifiers make more errors, their errors are more uniformly distributed. This can also be seen in the confusion matrices.

To explore further, we consider the effect of gender. We select the subset of female instances by left clicking in a histogram and intersect this with the African American subset (using the relationship widget). C3 has 0 recall on this subset of African American females, while other classifiers achieve more balanced performance. While these findings could have been found using scripting [FSV*19] or using specialized tools [CEH*19], Boxer can identify them using subset selection and comparative visualization.

5.6. Bias and Data Discovery: Literary Dating

In this use case, we again consider the TCP collection of historical documents (Section 5.2). We construct classifiers that determine whether a document is written after 1642 based on the 500 most common words in the corpus. While ground truth is known, effective classifiers can help understand how word usage changed at this critical date that marks the beginning of the English Civil War. The collection is skewed (only 25% of the documents were written before 1642). For the experiment, we took a random sample of 12,000 documents, and held out 30% using stratified sampling. While the testing set is balanced (1,800 per class), the training set is highly skewed (only 15% before 1642). We constructed a number of classifiers using various methods.

An image from an analysis session with Boxer is shown in Figure 7. On the training data, several classifiers achieve nearly perfect performance. However, the Parallel Metrics view (on the test set) shows that most classifiers provide high recall but low precision, suggesting that they were unable to successfully account for the class skew (in the training set). The SVM100W (a support vector machine using class weights and more regularization) provides the best performance in all metrics other than recall. The more balanced performance can also be seen in the Confusion Matrix grid. We focus on this classifier for our assessment.

We would expect that performance may be biased near the class boundary, as documents written near the boundary year may be similar to those on the other side (unless there was a dramatic change at the boundary). To check this effect, we select the errors of the classifier and view the selection in a histogram of dates (Figure 8). Normalizing this histogram (to account for the skewed distribution), confirms that most errors are in the buckets near the boundary. In contrast, the skewed classifiers generally made many errors in other buckets before 1642.

One explanation for the errors may be document length: near the civil war, many short documents were written (e.g., legal decrees). To explore this, we select the shortest documents and create a subset of documents written near the boundary. The skewed distributions in the lower left and center of Figure 7 make the result hard to interpret, but normalizing these histograms show that short documents are over-represented in the time period. However, intersecting the two sets (to select the short documents in the period) allows us to consider performance. Intersecting this set with the errors (and comparing with the total errors) show that prediction performance is better on short documents (Figure 8). This can also be seen in the Selection Performance view. Alternatively, we can select errors and period documents to see that there is a skew to-

Figure 7: *Boxer in the literature use case (Section 5.6). In the Classifier Performance view (top left), we see that* SVM(100W) *performs best in terms of accuracy. (1) From the Metrics Parallel and the Confusion Matrix views we learn that it is also the most balanced in terms of other metrics. (2) To assess whether classification is biased around the decision date (1642), we select the errors of the model (magenta), and view them on a date Histogram view and see that most errors happen in that period. (3) To test whether document length plays a role for these errors, we then select the shortest documents in the set in a document length Histogram view. (4) In the Performance Selection view (bottom right) reveals that shorter document (magenta) seem to be easier then longer ones.*

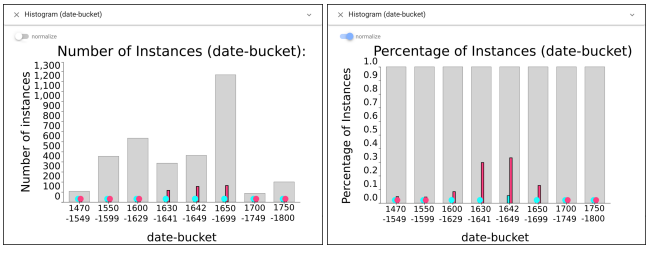

Figure 8: *Errors on all (magenta) and short documents (cyan) across temporal bins of documents. The right view is a normalized version of the left one. We can see that around 1642 there is a higher error rate in general, but the error rate on short documents is low.*

wards short documents. Using Boxer's ability to create subsets and compare them, we can examine details of classifier performance.

6. Discussion

We have presented a comprehensive approach for interactive comparison for machine learning classifier results, and a prototype implementation, Boxer. The approach combines subset identification, metric selection, and comparative visualization to enable detailed comparison of classifier results. We demonstrate the effectiveness of our approach for a range of datasets and model types through use cases.

6.1. Scalability

The classifier comparison problem can scale along all three "axes of hardness" [Gle18] with many instances, classifiers to compare, and complex relationships between classifiers. Boxer has been used to analyze experiments with up to tens of thousands of instances, a dozen classifiers, a dozen labels, and dozens of features. Boxer's in-browser JavaScript implementation grows sluggish at these scales. Providing interactive performance for larger data will require more efficient mechanisms to perform the set computations and will probably require sharing some of the computations in a back-end server. The more interesting scalability challenges relate to how well the Boxer design can handle larger problems.

The visual summaries used by Boxer are independent of instance quantity: they show aggregate quantities (e.g., counts). A bar chart appears the same if the Y axis represent 100 or a million. However, one challenge is dynamic range: small values in big sets can be important (e.g., identifying a few errors in a massive training set). Even at current scales, interesting bars may be a fraction of a pixel tall, and similar issues exist for color encodings. To combat these dynamic range issues, box designs use special encodings for small values. Zero (empty sets) are encoded differently than small sets,

and small sets are given special encodings that provide them with sufficient area to be clickable.

Instance views, such as lists, represent a different scalability challenge. With large numbers of instances, selections may contain many items. Presently, Boxer handles the performance issues of long lists by on-demand paging, and the usability of long lists by allowing for sorting and filtering. Future extensions could provide more automated assistance in finding interesting elements in long lists, such as representative subset sampling. In general, scalability concerns have led us to avoid per-instance displays, like the scatterplots used in Manifold [ZWM*19] or the stacks in ModelTracker [ACD*15].

Scaling to large numbers of classes and classifiers represent design challenges. The visual elements of bar charts and matrices must become smaller as the number of categories to show grows. For moderate numbers, matrix re-ordering techniques can help our designs remain useful for larger data. However, different visual designs may be required to afford comparisons between large numbers of features or classifiers. At present, Boxer treats features independently: the user must select specific features to examine. To better scale to large numbers of features, Boxer will need to incorporate interest operators to help guide users to interesting features.

6.2. Other Problem Types

Boxer is designed for the results of discrete choice classifiers. Extending the prototype and concepts to other problem types is important future work.

Probabilistic Classification Results: Boxer treats prediction probabilities as metadata features for analysis. This provides for limited analysis. Future extensions to handle probabilistic classifiers may adapt existing designs such as Squares [RAL*17] and Confusion Wheels [AHH*14], as well as standard metrics, such as ROC curves, and and area under the curve (AUC/ROC).

Continuous Prediction Problems: Presently, Boxer treats *regression* problems by discretizing the output into a set of discrete classes. Existing visualizations for assessing regression models (e.g., [MP13]) could be extended with the box abstractions.

Problems without Ground Truth: If "correct" labels are not known, Boxer selects a model as the "gold-standard." This limits application in unsupervised tasks. Supporting "differently correct" answers within the Boxer framework will require designs for better exploring agreement, perhaps by adapting designs for clustering assessment (e.g., [KEV*17, CD19]).

6.3. Limitations

Boxer presently uses a number of basic visualization designs. Other visualizations, such as scatterplots or non-linear dot plots [RW18], may be useful to better understand instance distributions. Adapting visualizations to work within Boxer requires identifying mechanisms for specifying selections as well as displaying overlaps with active selections. For example, with a scatterplot, overlap may be shown by coloring the dots and selection may be accomplished with brushing. However, computing the overlap to display new selections must be made efficient, and concise textual descriptions of brushing results are required to integrate into the selection management mechanisms.

Boxer presently is a stand alone tool for viewing experiment results. It provides no support for helping to design and run appropriate experiments. Better coupling with the experimental process offers opportunities as the increased facilities to analyze experiments suggests the potential for non-standard experiments. Similarly, Boxer is decoupled from the model building process. Integrating Boxer into an automated learning pipeline (e.g., [CHH*19, GHG*19, SCF*19, WMJ*19]) may provide a mechanism to more directly apply insights to improve models.

A key factor in Boxer is usability. The complexity of Boxer's interface is kept low by the choice of familiar visualizations and the use of a small set of basic abstractions that are used uniformly. However, in order to make complex analyses, a user must combine these basic elements in potentially complex ways. In principle, such complex assemblies are built up gradually from parts, but a user must know what combinations are possible and useful. To address these issues we plan to include pre-configured layouts to answer specific questions (as in [SSSG16]) and workflow-based guidance to suggest potential next steps to a user [CGM*17].

6.4. Conclusion

Despite these limitations, the Boxer prototype shows the potential for the approach to help users with comparing classifiers in machine learning experiment results. We are continuing to work with machine learning practitioners to refine the system and understand its potential. The key innovations of Boxer, the consistent use of the box abstraction, the use of multiple selections with set algebra, and the specific interface designs for connecting boxes and selections should be applicable in creating scalable systems for exploring comparisons of objects beyond machine learning classifiers.

Acknowledgements This work was funded in part by NSF Award 1841349 and DARPA award FA8750-17-2-010.

References

[AB16] ALAIN G., BENGIO Y.: Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv: 1610.01644* (Oct 2016). 2

[ACD*15] AMERSHI S., CHICKERING M., DRUCKER S., LEE B., SIMARD P., SUH J.: Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2015)* (April 2015). 2, 10

[AG16] ALEXANDER E., GLEICHER M.: Task-Driven Comparison of Topic Models. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (Jan 2016), 320–329. 3

[AHH*14] ALSALLAKH B., HANBURY A., HAUSER H., MIKSCH S., RAUBER A.: Visual Methods for Analyzing Probabilistic Classification Data. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (Dec 2014), 1703–1712. 3, 10

[AHM*17] ARRAS L., HORN F., MONTAVON G., MÜLLER K.-R., SAMEK W.: "What is relevant in a text document?": An interpretable machine learning approach. *PLOS ONE 12*, 8 (Aug 2017). 2

[AL20] AHN Y., LIN Y.: Fairsight: Visual analytics for fairness in decision making. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan 2020), 1086–1095. 2

[ALMK16] ANGWIN J., LARSON J., MATTU S., KIRCHNER L.: Machine bias. https://www.propublica.org/article/, 2016. 7

[AMA*14] ALSALLAKH B., MICALLEF L., AIGNER W., HAUSER H., MIKSCH S., RODGERS P.: Visualizing Sets and Set-typed Data: State-of-the-Art and Future Challenges. In *Eurovis STAR Reports* (2014), The Eurographics Association. 3

[AR15] AUBRY M., RUSSELL B. C.: Understanding Deep Features with Computer-Generated Imagery. In *2015 IEEE International Conference on Computer Vision (ICCV)* (Dec 2015), IEEE, pp. 2875–2883. 2

[Ber10] BERTIN J.: *Semiology of Graphics*, 2nd ed. ESRI Press, Redlands, CA, 2010. 2

[Bre01] BREIMAN L.: Random Forests. *Machine Learning 45*, 1 (2001), 5–32. 2

[CBS*16] CHOI E., BAHADORI M. T., SUN J., KULAS J., SCHUETZ A., STEWART W.: Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems* (2016), pp. 3504–3512. 2

[CD19] CAVALLO M., DEMIRALP Ç.: Clustrophile 2: Guided visual clustering analysis. *IEEE transactions on visualization and computer graphics 25*, 1 (2019), 267–276. 3, 10

[CEH*19] CABRERA A. A., EPPERSON W., HOHMAN F., KAHNG M., MORGENSTERN J., CHAU D. H.: Fairvis: Visual analytics for discovering intersectional bias in machine learning. *IEEE Conference on Visual Analytics Science and Technology (VAST)* (2019). 2, 7, 8

[CGM*17] CENEDA D., GSCHWANDTNER T., MAY T., MIKSCH S., SCHULZ H.-J., STREIT M., TOMINSKI C.: Characterizing Guidance in Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan 2017), 111–120. 10

[CHH*19] CASHMAN D., HUMAYOUN S. R., HEIMERL F., PARK K., DAS S., THOMPSON J., SAKET B., MOSCA A., STASKO J., ENDERT A., GLEICHER M., CHANG R.: A user-based visual analytics workflow for exploratory model analysis. *Computer Graphics Forum 38*, 3 (2019), 185–199. 10

[CLG*15] CARUANA R., LOU Y., GEHRKE J., KOCH P., STURM M., ELHADAD N.: Intelligible Models for HealthCare. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15* (New York, New York, USA, 2015), ACM Press, pp. 1721–1730. 2

[Cra96] CRAVEN M.: *Extracting Comprehensible Models from Trained Neural Networks*. Ph. D. dissertation, University of Wisconsin - Madison, 1996. 2

[CS68] CHEN C. F., SHIEH L. S.: A novel approach to linear model simplification. *International Journal of Control 8*, 6 (Dec 1968), 561–570. 2

[CS96] CRAVEN M. W., SHAVLIK J. W.: Extracting tree-structured representations of trained neural networks. In *Advances in Neural Information Processing Systems* (1996), vol. 8, pp. 24–30. 2

[DWOB20] DASGUPTA A., WANG H., O'BRIEN N., BURROWS S.: Separating the wheat from the chaff: Comparative visual cues for transparent diagnostics of competing models. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan 2020), 1043–1053. 3, 5

[EF10] ELMQVIST N., FEKETE J.-D.: Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE transactions on visualization and computer graphics 16*, 3 (2010), 439–54. 4

[FSV*19] FRIEDLER S. A., SCHEIDEGGER C., VENKATASUBRAMANIAN S., CHOUDHARY S., HAMILTON E. P., ROTH D.: A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (New York, NY, USA, 2019), FAT* '19, ACM, pp. 329–338. 2, 7, 8

[GHG*19] GIL Y., HONAKER J., GUPTA S., MA Y., D'ORAZIO V., GARIJO D., GADEWAR S., YANG Q., JAHANSHAD N.: Towards human-guided machine learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces - IUI '19* (Marina del Ray, California, 2019), ACM Press, pp. 614–624. 10

[Gle13] GLEICHER M.: Explainers: Expert explorations with crafted projections. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec 2013), 2042–2051. 2, 6

[Gle16] GLEICHER M.: A Framework for Considering Comprehensibility in Modeling. *Big Data 4*, 2 (Jun 2016), 75–88. 2

[Gle18] GLEICHER M.: Considerations for Visualizing Comparison. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan 2018), 413–423. 9

[HG18] HEIMERL F., GLEICHER M.: Interactive analysis of word vector embeddings. *Computer Graphics Forum 37*, 3 (2018), 253–265. 3

[HPB*14] HENELIUS A., PUOLAMÄKI K., BOSTRÖM H., ASKER L., PAPAPETROU P.: A peek into the black box: exploring classifiers by randomization. *Data Mining and Knowledge Discovery 28*, 5-6 (Sep 2014), 1503–1529. 2

[HPRP20] HOHMAN F., PARK H., ROBINSON C., POLO CHAU D. H.: Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan 2020), 1096–1106. 2

[IK11] ISHIZAKI S., KAUFER D.: DocuScope: Computer-aided rhetorical analysis. In *Applied Natural Language Processing and Content Analysis: Advances in Identification, Investigation, and Resolution*, McCarthy P., Boonthum C., (Eds.). IGI Global, 2011. 6

[KDS*17] KRAUSE J., DASGUPTA A., SWARTZ J., APHINYANAPHONGS Y., BERTINI E.: A workflow for visual diagnostics of binary classifiers using instance-level explanations. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2017), IEEE, pp. 162–172. 2

[KEV*17] KWON B. C., EYSENBACH B., VERMA J., NG K., DEFILIPPI C., STEWART W. F., PERER A.: Clustervision: Visual Supervision of Unsupervised Clustering. *IEEE Transactions on Visualization and Computer Graphics* (2017), 1–1. 3, 10

[KLTH10] KAPOOR A., LEE B., TAN D., HORVITZ E.: Interactive optimization for steering machine classification. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10* (New York, New York, USA, Apr 2010), ACM Press, p. 1343. 3

[LCGH13] LOU Y., CARUANA R., GEHRKE J., HOOKER G.: Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13* (New York, New York, USA, 2013), ACM Press, p. 623. 2

[Lip16] LIPTON Z. C.: The Mythos of Model Interpretability. *arXiv preprint arXiv:1606.03490* (Jun 2016), 1606.03490. 2

[LSC*18] LIU M., SHI J., CAO K., ZHU J., LIU S.: Analyzing the training processes of deep generative models. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan 2018), 77–87. 2

[LSC19] LEE K., SOOD A., CRAVEN M.: Understanding Learned Models by Identifying Important Features at the Right Resolution. In *AAAI* (Nov 2019). 2, 6

[LSL*17] LIU M., SHI J., LI Z., LI C., ZHU J., LIU S.: Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan 2017), 91–100. 2

[MBVV07] MARTENS D., BAESENS B., VAN GESTEL T., VANTHIENEN J.: Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research 183*, 3 (Dec 2007), 1466–1476. 2

[MCZ*17] MING Y., CAO S., ZHANG R., LI Z., CHEN Y., SONG Y., QU H.: Understanding hidden memories of recurrent neural networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)* (Oct 2017), pp. 13–24. 2

[MLMP18] MUHLBACHER T., LINHARDT L., MOLLER T., PIRINGER H.: TreePOD: Sensitivity-Aware Selection of Pareto-Optimal Decision Trees. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan 2018), 174–183. 2

[MMD*19] MURUGESAN S., MALIK S., DU F., KOH E., LAI T. M.: Deepcompare: Visual and interactive comparison of deep learning model performance. *IEEE Computer Graphics and Applications 39*, 5 (Sep 2019), 47–59. 2

[MP13] MÜHLBACHER T., PIRINGER H.: A Partition-Based Framework for Building and Validating Regression Models. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec 2013), 1962–1971. 3, 10

[MQB19] MING Y., QU H., BERTINI E.: Rulematrix: Visualizing and understanding classifiers with rules. *IEEE transactions on visualization and computer graphics 25*, 1 (2019), 342–352. 2

[MV16] MAHENDRAN A., VEDALDI A.: Visualizing Deep Convolutional Neural Networks Using Natural Pre-images. *International Journal of Computer Vision 120*, 3 (Dec 2016), 233–255. 2

[MXC*20] MING Y., XU P., CHENG F., QU H., REN L.: Protosteer: Steering deep sequence model with prototypes. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan 2020), 238–248. 3

[MXLM20] MA Y., XIE T., LI J., MACIEJEWSKI R.: Explaining vulnerabilities to adversarial machine learning through visual analytics. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan 2020), 1075–1085. 2

[OJ02] OLDEN J. D., JACKSON D. A.: Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling 154*, 1-2 (Aug 2002), 135–150. 2

[OW16] OBERMANN L., WAACK S.: Interpretable Multiclass Models for Corporate Credit Rating Capable of Expressing Doubt. *Frontiers in Applied Mathematics and Statistics 2* (Oct 2016). 2

[Pow11] POWERS D.: Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correllation. *Journal of Machine Learning Technologies 2*, 1 (2011), 37–63. 3

[RAL*17] REN D., AMERSHI S., LEE B., SUH J., WILLIAMS J. D.: Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan 2017), 61–70. 3, 10

[RFFT17] RAUBER P. E., FADEL S. G., FALCAO A. X., TELEA A. C.: Visualizing the Hidden Activity of Artificial Neural Networks. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan 2017), 101–110. 2

[RSG16] RIBEIRO M. T., SINGH S., GUESTRIN C.: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16* (New York, New York, USA, Feb 2016), ACM Press, pp. 1135–1144. 2

[RW18] RODRIGUES N., WEISKOPF D.: Nonlinear dot plots. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan 2018), 616–625. 10

[SAMG14] SARIKAYA A., ALBERS D., MITCHELL J., GLEICHER M.: Visualizing Validation of Protein Surface Classifiers. *Computer Graphics Forum 33*, 3 (Jun 2014), 171–180. 3

[SCF*19] SANTOS A., CASTELO S., FELIX C., ONO J. P., YU B., HONG S., SILVA C. T., BERTINI E., FREIRE J.: Visus: An Interactive System for Automatic Machine Learning Model Building and Curation. In *2019 Workshop on Human-In-the-Loop Data Analytics (HILDA'19)* (Jul 2019). 10

[SDV*16] SELVARAJU R. R., DAS A., VEDANTAM R., COGSWELL M., PARIKH D., BATRA D.: Grad-CAM: Why did you say that? *arXiv preprint arXiv:1611.07450* (Nov 2016). 2

[SDW09] SLINGSBY A., DYKES J., WOOD J.: Configuring Hierarchical Layouts to Address Research Questions. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov 2009), 977–984. 4

[SGB*19] STROBELT H., GEHRMANN S., BEHRISCH M., PERER A., PFISTER H., RUSH A. M.: Seq2seq-vis: A visual debugging tool for sequence-to-sequence models. *IEEE transactions on visualization and computer graphics 25*, 1 (2019), 353–363. 3

[SGK17] SHRIKUMAR A., GREENSIDE P., KUNDAJE A.: Learning Important Features Through Propagating Activation Differences. *PMLR 70* (Apr 2017), 3145–3153. 2

[SGPR18] STROBELT H., GEHRMANN S., PFISTER H., RUSH A. M.: Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan 2018), 667–676. 2

[ŠK14] ŠTRUMBELJ E., KONONENKO I.: Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems 41*, 3 (Dec 2014), 647–665. 2

[SSSE20] SPINNER T., SCHLEGEL U., SCHÄFER H., EL-ASSADY M.: explainer: A visual analytics framework for interactive and explainable machine learning. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan 2020), 1064–1074. 2

[SSSG16] SZAFIR D. A., STUFFER D., SOHAIL Y., GLEICHER M.: TextDNA: Visualizing Word Usage with Configurable Colorfields. *Computer Graphics Forum 35*, 3 (Jun 2016), 421–430. 10

[STY17] SUNDARARAJAN M., TALY A., YAN Q.: Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR.org, pp. 3319–3328. 2

[SVC*10] STEYERBERG E. W., VICKERS A. J., COOK N. R., GERDS T., GONEN M., OBUCHOWSKI N., PENCINA M. J., KATTAN M. W.: Assessing the performance of prediction models: a framework for traditional and novel measures. *Epidemiology (Cambridge, Mass.) 21*, 1 (Jan 2010), 128–38. 3

[SVZ13] SIMONYAN K., VEDALDI A., ZISSERMAN A.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv preprint arXiv:1312.6034* (Dec 2013). 2

[Sza18] SZAFIR D. A.: Modeling Color Difference for Visualization Design. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan 2018), 392–401. 4

[TKDB17] TAMAGNINI P., KRAUSE J., DASGUPTA A., BERTINI E.: Interpreting Black-Box Classifiers Using Instance-Level Visual Explanations. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics - HILDA'17* (New York, New York, USA, 2017), ACM Press, pp. 1–6. 2

[TLKT09] TALBOT J., LEE B., KAPOOR A., TAN D. S.: EnsembleMatrix. In *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09* (New York, New York, USA, Apr 2009), ACM Press, p. 1283. 3

[Wal84] WALLIS, JEROLD W AND SHORTLIFFE E. H.: Customized explanations using causal knowledge. In *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA, 1984, pp. 371–388. 2

[Wei80] WEINER J.: BLAH, a system which explains its reasoning. *Artificial Intelligence 15*, 1-2 (Nov 1980), 19–48. 2

[WFH11] WITTEN I. H., FRANK E., HALL M. A.: *Data Mining: Practical Machine Learning Tools and Techniques, 3e*. Morgan Kaufmann, Burlington, 2011. 3

[WH10] WITMORE M., HOPE J.: The Hundredth Psalm to the Tune of "Green Sleeves": Digital Approaches to Shakespeare's Language of Genre. *Shakespeare Quarterly 61*, 3 (2010), 357–390. 6

[Wil05] WILKINSON L.: *The Grammar of Graphics, Second Edition*, 2nd ed. Statistics and Computing. Springer-Verlag, New York, 2005. 2

[WMJ*19] WANG Q., MING Y., JIN Z., SHEN Q., LIU D., SMITH

M. J., VEERAMACHANENI K., QU H.: ATMSeer: Increasing Transparency and Controllability in Automated Machine Learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19* (Glasgow, Scotland Uk, 2019), ACM Press, pp. 1–12. 10

[WPB*20] WEXLER J., PUSHKARNA M., BOLUKBASI T., WATTENBERG M., VIÉGAS F., WILSON J.: The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan 2020), 56–65. 2

[WR15] WANG F., RUDIN C.: Falling Rule Lists. In *18th International Conference on Artificial Intelligence and Statistics (AISTATS)* (Nov 2015). 2

[WSW*17] WONGSUPHASAWAT K., SMILKOV D., WEXLER J., WILSON J., MANE D., FRITZ D., KRISHNAN D., VIEGAS F. B., WATTENBERG M.: Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2017), 1–12. 2

[YCN*15] YOSINSKI J., CLUNE J., NGUYEN A., FUCHS T., LIPSON H.: Understanding Neural Networks Through Deep Visualization. In *Deep Learning Workshop, 31 st International Conference on Machine Learning* (2015). 2

[YXX*19] YE X., XIANG S., XIA J., WU J., CHEN Y., LU S.: Interactive correction of mislabeled training data. *IEEE Conference on Visual Analytics Science and Technology (VAST)* (2019). 2

[ZF14] ZEILER M., FERGUS R.: Visualizing and Understanding Convolutional Networks. In *ECCV 2014* (Cham, 2014), Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), vol. 8689 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 818–833. 2

[ZWM*19] ZHANG J., WANG Y., MOLINO P., LI L., EBERT D. S.: Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE transactions on visualization and computer graphics 25*, 1 (2019), 364–373. 2, 10