SliceLens: Guided Exploration of Machine Learning Datasets

Daniel Kerrigan
kerrigan.d@northeastern.edu
Northeastern University
Boston, Massachusetts, USA

Enrico Bertini
e.bertini@northeastern.edu
Northeastern University
Boston, Massachusetts, USA

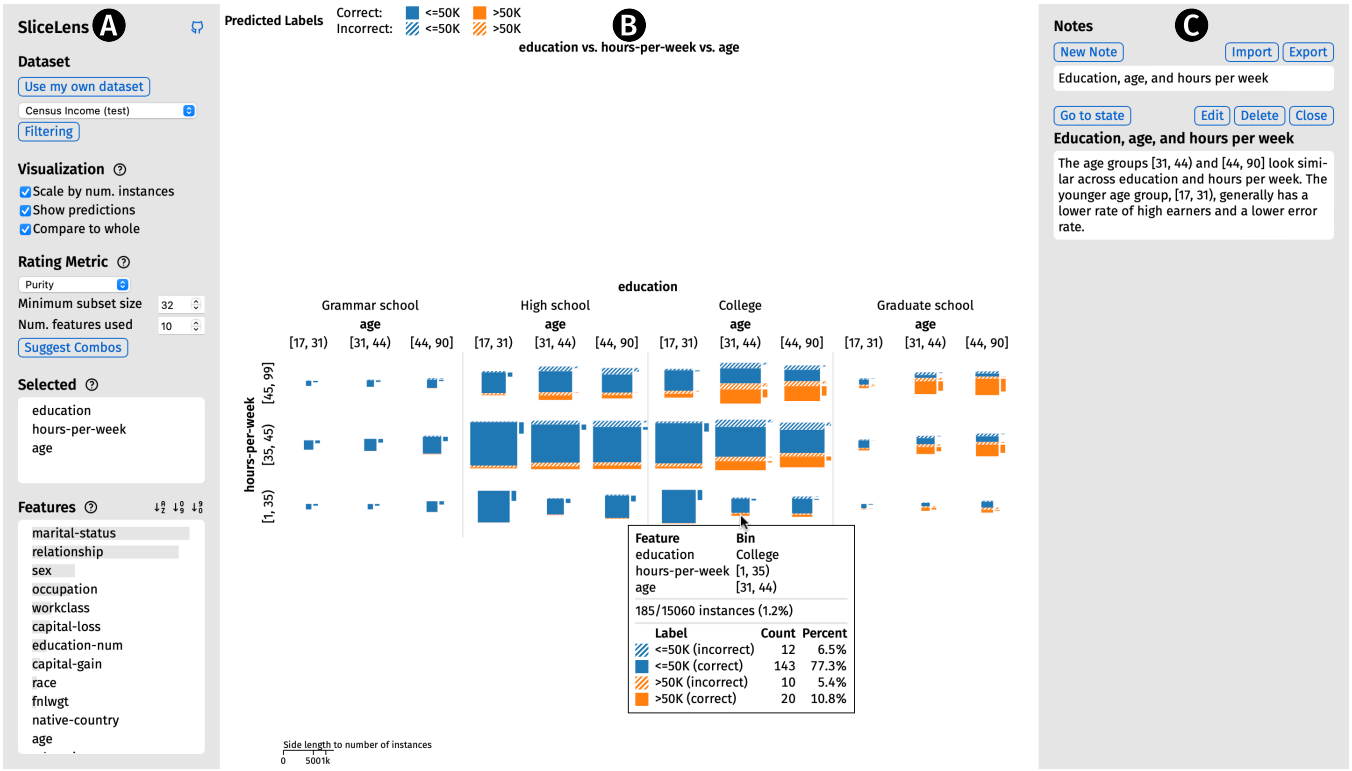


Figure 1: SliceLens on the Census Income (Adult) dataset. A) The left sidebar contains the controls for the visualization. B) The main component is a visualization of the intersections of feature bins. Each square visualizes the label distribution of a subset of the data. C) The right sidebar is for notes, which lets users document and revisit their findings.

ABSTRACT

SliceLens is a tool for exploring labeled, tabular, machine learning datasets. To explore a dataset, the user selects combinations of features in the dataset that they are interested in. The tool splits those features into bins and then visualizes the label distributions for the subsets of data created by the intersections of the bins. SliceLens guides the user in determining which feature combinations to explore. Guidance is based on a user-selected rating metric, which assigns a score to the subsets created by a given combination of features. The purpose of the metrics are to detect interesting patterns in the subsets, such as subsets that have high label purity or an uneven distribution of errors. SliceLens uses the metrics to guide the user towards combinations of features that create potentially interesting subsets in two ways. First, SliceLens assigns a rating to each feature based on the subsets that would be created by selecting that feature. This incremental guidance can help the user determine which feature to select next. Second, SliceLens can suggest combinations of features ranked according to the chosen metric, which the user can then cycle through.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Human-centered computing** → **Visualization systems and tools**; **Visual analytics**.

KEYWORDS

machine learning, visualization, visual analytics, guidance, exploration

ACM Reference Format:
Daniel Kerrigan and Enrico Bertini. 2018. SliceLens: Guided Exploration of Machine Learning Datasets. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

When exploring a labeled, tabular dataset that is used to train or evaluate a machine learning model, the relationship between features and labels is of key interest, since the features are the inputs to the model and the labels are the desired or actual outputs of the model. When both ground truth and predicted labels are available, identifying where the model makes a high or low amount of errors is also important. Both exploring the relationship between features and labels and exploring where a model tends to make errors can be supported by dividing the instances in the dataset into subsets based on their values for specific features of interest. For example, a data scientist may be exploring the Census Income (Adult) dataset [4] and want to understand the impact that a person's age and education has on their income, which is represented as a binary label of whether the person makes more or less than $50,000 in a year. They can divide the dataset into subsets, such as young people with high education, old people with high education, young people with low education, etc. By comparing the distribution of labels in each subset, they can get a sense of the relationship between age, education, and income. Likewise, once the data scientist has trained a model, they may be interested in comparing the model's error rates between these subsets in order to gain an understanding of where the model is inaccurate.

This type of exploration can be continued by examining the subsets created by different feature combinations. One difficulty with this approach is that, in many datasets, there are too many possible combinations of features to be able to exhaustively explore all of them. In these cases, the data scientist will necessarily have to focus on specific feature combinations, but they may not know which ones are worth their attention.

To address this problem, we present SliceLens, which is a visualization tool that guides users in exploring labeled, tabular, machine learning datasets. Data exploration in SliceLens is driven by the user selecting combinations of features. The tool bins the selected features and visualizes the subsets of the data created by the intersections of the bins; thus providing a simple interface for the user to quickly explore relationships between combinations of features and the labels. SliceLens provides guidance to the user to help them determine which combinations of features to explore. Guidance in SliceLens is based on a user chosen rating metric, which assigns a score to the subsets created by a given combination of features. These metrics are used to identify feature combinations that result in subsets with potentially interesting patterns in their label distributions, such as subsets that effectively group instances by their label or subsets that have uneven error distributions. SliceLens offers two forms of guidance. The first is feature ratings. Using the chosen metric, SliceLens assigns a rating to each feature based on the subsets that would be created by adding that feature to the visualization. The feature ratings are updated as the user adds and removes features from the visualization. The feature ratings provide incremental guidance and leave the user in control of what features they select, allowing them to factor in their own interests. The second form of guidance that SliceLens offers is by suggesting feature combinations. SliceLens ranks combinations of one to three features according to the chosen metric and enables the user to easily cycle through the suggested combinations, thereby offering a more automated form of guidance. SliceLens can be used to explore both classification and regression datasets. These datasets can either have only ground truth labels or have both ground truth and predicted labels.

2 RELATED WORK

The visualization and interaction that SliceLens provides is similar to Facets Dive [5], which is a visualization tool for exploring machine learning datasets and is a part of the What-If Tool [14]. Like SliceLens, Facets Dive allows users to explore datasets by selecting combinations of features and the tool then splits the dataset into subsets and creates visualizations of the subsets. Facets Dive visualizes each instance in the dataset, whereas SliceLens visualizes the label distribution in each subset. A significant difference between SliceLens and Facets Dive is that Facets Dive does not guide or assist the user in determining what combinations of features to explore. The user is on their own in deciding what visualizations to look at. The guidance that SliceLens offers the user through the feature ratings and feature combination suggestions is a key contribution.

There are many tools designed for visually analyzing the results of a machine learning model. Squares [13] visualizes the performance of multi-class classification models, but it does not incorporate feature-level information and does not support regression models. Slice Finder [11] identifies data subsets where model performance is poor, however, its only visualization is an interactive scatter plot showing the size and effect size of the identified subsets. It does not visualize the predictions in the subsets. Visual Auditor [10] builds additional visualizations on top of the Slice Finder algorithm that are designed for analyzing model bias, but do not show label distributions of the subsets. MLCube Explorer [9] lets the user compare the performance of different models across user-defined subsets of the data. Unlike SliceLens, MLCube Explorer does not guide the user in determining which subsets to explore. FairVis [2] is particularly related to SliceLens, as they both seek to help the user determine which subsets to explore when it is not practical to explore them all. Similar to SliceLens, FairVis suggests subsets to the user, which can be sorted based on different fairness metrics. The suggested subsets are generated through K-means clustering, which means that the subsets are not strictly defined by specific feature values, as they are in SliceLens. FairVis places primary emphasis on comparing subsets according to metrics such as precision, recall, and accuracy, whereas SliceLens focuses on comparing the subsets created by specific feature combinations according to their label distributions. FairVis has limited support for this type of comparison, since the user can compare the ground truth labels between only two subsets at once. In addition, FairVis is limited to binary classification datasets, whereas SliceLens also supports multi-class and regression datasets. SliceTeller [15] identifies and visualizes under-performing subsets and estimates the impact of prioritizing those subsets in model training. All of the tools referenced in this paragraph are for analyzing the results

and performance of a model and therefore require that the model already be trained. They do not support guiding the exploration of training datasets before a model exists, like SliceLens does. For example, a data scientist might use SliceLens to explore a dataset before training to familiarize themselves with it and to get a sense for the key relationships between the features and labels and then return to SliceLens once the model is trained to explore its predictions and the errors that it makes.

3 INTERFACE

3.1 Overview

Users explore tabular datasets in SliceLens by selecting combinations of features that they are interested in. SliceLens then splits the dataset into subsets according to the selected features. This is done by binning the values in each feature. By default, quantitative features are split into three equal-width bins. Categorical features have one bin per category. If the feature has more than five categories, then there is one bin for each of the top four categories and the remaining categories are grouped in an "Other" bin. The user can edit the default bins for a feature. The data subsets are created by the intersections of the bins for the selected features. The tool then produces a visualization showing how the labels distribute across the generated data subsets.

For example, Figure 1 shows the Census Income (Adult) dataset with the age, number of hours worked per week, and education level features selected. In this case, the age and number of hours worked per week features both have three bins and the education feature has four bins. The result is a visualization that shows the subsets for all 36 possible combinations of the bins for age, number of hours worked per week, and education level. In SliceLens, a maximum of four features can be selected at once. This restriction is supported by Halford et al., whose findings suggest that humans can effectively process the interactions between no more than four variables at a single time [7].

The user interface for SliceLens contains three primary parts (Figure 1). The main section of the tool is a visualization of the selected features (Figure 1B). The left sidebar contains the controls for the visualization (Figure 1A). The right sidebar allows the user to take notes and save states of the visualization (Figure 1C).

The center section of SliceLens (Figure 1B) contains a visualization of the data subsets that are created by the intersections of the selected features' bins. The visualization always includes every instance in the dataset by default, though the user can filter instances according to specific feature values. The label distribution in each subset is visualized with a square. If one feature is selected, then the squares are arranged horizontally. If two features are selected, then the squares are arranged in a matrix, where the columns are the bins of the first selected feature and the rows are the bins of the second selected feature, as shown in Figure 2. Adding a third feature creates nested columns. For example, you can see a hierarchy of columns in Figure 1. The first feature, education, is at the top of the column hierarchy. There is one top-level column for each bin of the education feature (Grammar school, High school, College, and Graduate school). Each of these top-level columns is divided into three columns for the bins of the age feature (17-31, 31-44, and 44-90). When a fourth feature is selected, nested rows are created in a similar manner. The subset visualizations are discussed in more detail in Section 3.2 and Section 3.3.

At the top of the left sidebar (Figure 1A), the user can import their dataset in CSV format. The bottom of the left sidebar contains a list of the currently selected features and a list of all of the features in the dataset. These lists enable the user to select, remove, and re-order features in the visualization. Clicking a feature's edit icon brings up a modal that lets the user control how the feature is split into bins. For categorical features, the user can group the feature values into any number of bins and rename and sort the bins. For quantitative features, the user can specify the number of bins and choose between equal-width, equal-frequency, and custom user-specified bins. Equal-width bins divide the feature's domain into intervals that have the same size. Equal-frequency bins use quantiles to split the feature's values, which attempts to put roughly the same number of instances into each bin.

The right sidebar is for note taking (Figure 1C). Users can create notes and link them to the state of the visualization. This allows users to revisit their findings in the visualization at a later point or to export them in order to share the findings with others.

SliceLens is an open-source[1] web application built using Svelte [8] and D3 [1].

3.2 Classification Dataset Visualization

SliceLens supports both classification and regression datasets. Figure 1 shows predicted labels for the binary Census Income dataset. For classification datasets, each subset is represented by a square. Each square has one layer for each class in the dataset, where the height of a layer is proportional to the number of instances in that subset with that class. The layers are colored according to their class. The user can choose whether they visualize the ground truth class labels or the predicted class labels, if available. When color encodes predicted labels, stripes represent incorrect predictions. The number of classes that SliceLens supports is limited by the use of color hue to distinguish between the classes in the visualization, therefore it works best with binary or few-class datasets. By default, the area of the square encodes the number of instances in the subset. The user can choose to make all of the squares the same size in order to make comparing distributions easier. Hovering over a square shows a tooltip, which provides details about the data subset (Figure 1B).

We considered representing each subset with a bar chart instead of with a square. One benefit of bar charts is that bar length is a more precise encoding than area [3]. However, we see a few issues with using bar charts that lead us to stick with the squares. First, the bars must be oriented either vertically or horizontally. When multiple features are selected, the subset visualizations are arranged in a grid. Having horizontally or vertically oriented bars would promote a specific reading direction on the data (e.g., horizontal bar charts would promote reading columns of subsets over rows). Choosing to orient the layers in the squares either horizontally or vertically also promotes a specific reading direction, but we believe that the centrally aligned squares lessens this effect. Second, despite being more precise, we find that the bar charts scale less effectively than the squares, particularly when a subset contains a small percent

[1] https://github.com/nyuvis/SliceLens

of the dataset. Lastly, we believe that the squares communicate the part-to-whole relationship of the labels in a subset better than the bar charts. With bar charts, the user may be more inclined to compare specific bars between subsets rather than comparing the overall distributions.

For classification datasets, the user has the option to highlight how the distribution of each subset differs from the distribution of the entire dataset (Figure 1B). If a subset has a higher percentage of a given class than the dataset as a whole, then there is a bar to the right of that class's layer in the square. The height of the bar encodes the percentage point increase of the class in the subset compared to the entire dataset. When visualizing a group of subsets, these bars make it easier to tell which subsets have a relative increase or decrease for a given class when compared to the entire dataset. When analyzing a group of subsets without showing these bars, the user would have to remember the label distribution of the whole dataset and compare that to the distribution of each subset in order to understand how they differ. This becomes easier when the difference is directly shown with the bars. They are particularly helpful when the dataset has imbalanced labels.

3.3 Regression Dataset Visualization

Figure 2 contains visualizations of the Bike Rentals regression dataset [6, 4]. As with classification datasets, each subset of a regression dataset is visualized by a square, where the size of the square represents the number of instances in the subset and the layers of the square show the label distribution. When showing ground truth labels (Figure 2A), the continuous labels are discretized into approximately twenty equal-width bins. To help better visualize skewed distributions, the user may also choose to use equal-frequency bins instead. A sequential color scale is used to visualize the distribution of the binned labels. If the user chooses to show predictions (Figure 2B), then SliceLens calculates the difference between the ground truth label and the predicted label for each instance. These delta values are then similarly discretized into approximately twenty bins. A diverging color scale is used to visualize the distribution of the binned delta values in order to differentiate between over and under predictions.

We considered using histograms to visualize the subsets rather than squares. Our reasons for preferring squares over bar charts for classification datasets also apply to preferring squares over histograms for regression datasets. One additional reason relates to space efficiency. A histogram includes space for each bin regardless of whether or not that bin is empty. The squares, however, do not use any space to show empty bins.

4 GUIDANCE

In a dataset with twenty features, there are 1350 unique combinations of one to three features. This exemplifies that exhaustively exploring the subsets created by all possible combinations of features is not practical. Therefore, the user will necessarily focus on certain combinations of features. To help the user determine which combinations of features are worth focusing on, SliceLens offers two forms of guidance: feature ratings and suggested feature combinations. The goal of the guidance is to direct the user towards combinations of features that result in subsets with interesting label distributions. To do this, both forms of guidance use a user-selected rating metric to assign a score to the subsets created by a given combination of features. The metrics seek to capture potentially interesting patterns in the subsets' labels.

4.1 Rating Metrics

The purpose of a rating metric is to assign a score to the set of subsets that are created by a given combination of features. The metrics are designed to give a higher rating to feature combinations that create subsets that have label distributions that the user may potentially find to be interesting. For example, when analyzing a training dataset that has ground truth labels, we anticipate that users will be interested in finding combinations of features whose subsets effectively group together instances with the same label (or similar labels in the case of regression datasets), as this could indicate that the features may be particularly informative for a model's prediction. When analyzing a validation dataset that has both ground truth and predicted labels, we anticipate that users will be interested in where the model makes errors and how those errors distribute across certain subsets. For example, they may be interested in identifying feature combinations where the model makes more or worse errors in one subset than another. These are the types of patterns that we seek to capture through the rating metrics. SliceLens has six predefined rating metrics that the user can choose from, which are split between metrics for classification datasets and metrics for regression datasets. Only a single rating metric is used for guidance at a time. The input to each metric is the set of subsets that are created by a given feature combination. In order to prevent the rating metrics from being overly influenced by subsets that are too small to be significant, the user can set a subset size threshold to filter out small subsets from the calculations.

4.1.1 Classification Metrics. For classification datasets, the user can choose between one of four metrics: *purity*, *error deviation*, *error count*, and *error percent*.

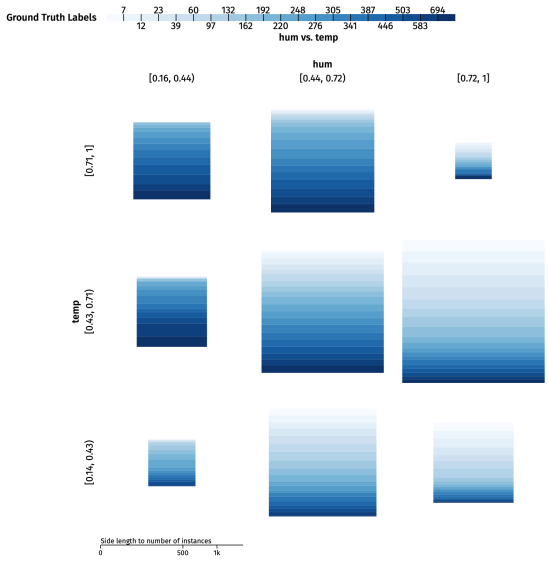
The *purity* metric is useful for guiding users to features that do better jobs at separating the instances into subsets by their ground truth label. The metric gives higher ratings to subsets that have a lower weighted average entropy, based only on the ground truth labels. This is similar to how entropy is used in building decision trees when deciding how to best split a given node [12].

We define a dataset D to be a set of N labeled instances. Let C be the set of unique labels in D. For a given subset $S \subset D$, we define the entropy of S in (1), where p_c is the percent of instances in S whose ground truth label is c.

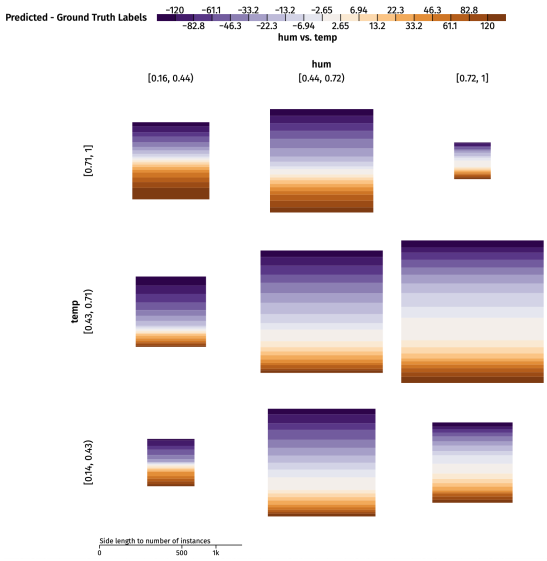
$$\text{entropy}(S) = -\sum_{c \in C} p_c \log_2 p_c \tag{1}$$

If we split D into a set of disjoint subsets T, then we define the *purity* metric for the set of subsets in (2). A lower weighted average entropy indicates that the subsets are more pure, therefore we negate this value so that a higher rating corresponds to purer subsets.

$$\text{purity}(T) = -\sum_{S \in T} \frac{|S|}{N} \text{entropy}(S) \tag{2}$$



(A) Ground truth labels.



(B) Difference between ground truth and predicted labels.

Figure 2: Visualizations of subsets created by the temperature and humidity features in the Bike Rentals dataset, using equal-frequency binning for the color scales. The target label in this dataset is the number of bikes rented from a bike share system.

The *error deviation* metric seeks to identify sets of subsets that have uneven distribution of errors. An even distribution of errors across the subsets would mean that each subset has the same percentage of errors. With an uneven distribution of errors, some subsets have a disproportionately high or low amount of errors. The *error deviation* metric gives higher ratings to sets of subsets with higher standard deviation of percent error. For a given set of subsets, the *error deviation* metric calculates the error rate for each subset and then takes their standard deviation.

In (3), we define a function that counts the number of errors in a subset. Assume that "ground" and "predicted" are functions that return the ground truth label and predicted label of an instance.

$$\text{numErrors}(S) = \sum_{x \in S} [\text{ground}(x) \neq \text{predicted}(x)] \tag{3}$$

We can then define the *error deviation* metric in (4), where σ is the standard deviation function.

$$\text{errorDeviation}(T) = \sigma(\{\frac{\text{numErrors}(S)}{|S|} | S \in T\}) \tag{4}$$

The *error count* and *error percent* metrics can guide users towards an individual subset that has a high number or percent of errors. The *error count* metric computes the number of errors in each subset and takes the maximum value (5). Similarly, the *error percent* metric computes the error rate for each subset and takes the maximum value (6).

$$\text{errorCount}(T) = \max(\{\text{numErrors}(S) | S \in T\}) \tag{5}$$

$$\text{errorPercent}(T) = \max(\{\frac{\text{numErrors}(S)}{|S|} | S \in T\}) \tag{6}$$

4.1.2 Regression Metrics. For regression datasets, the user can choose between one of two metrics: *similarity* and *MSE deviation*.

The *similarity* metric guides users in finding subsets that group instances with similar labels, akin to the *purity* metric for classification datasets. The *similarity* metric calculates the standard deviation of the ground truth labels in each subset and takes their weighted average (7, 8). A low standard deviation of the ground truth labels in a subset indicates that the subset contains instances with similar labels. For example, the subset might contain instances that all have low label values. In contrast, a subset with a higher standard deviation may contain both instances with low label values and high label values. There is a leading negative sign in (8) so that lower deviations correspond to higher similarity ratings.

$$\text{labelStdDev}(S) = \sigma(\{\text{ground}(x) | x \in S\}) \tag{7}$$

$$\text{similarity}(T) = -\sum_{S \in T} \frac{|S|}{N} \text{labelStdDev}(S) \tag{8}$$

Similar to the *error deviation* metric for classification datasets, the *MSE deviation* metric guides users towards sets of subsets that have uneven error distributions. That is, it can help the user find feature combinations that result in subsets where some subsets have worse errors than others. To calculate the *MSE deviation* metric for a given set of subsets, SliceLens computes the mean-squared error for each subset and takes their standard deviation, as in (9) and (10).

$$\text{MSE}(S) = \frac{1}{|S|} \sum_{x \in S} (\text{predicted}(x) - \text{ground}(x))^2 \tag{9}$$

$$\text{mseDeviation}(T) = \sigma(\{\text{MSE}(S) | S \in T\}) \tag{10}$$

4.2 Feature Ratings

The first form of guidance that SliceLens offers is assigning a rating to each feature. The rating for a feature is calculated according to the chosen rating metric using the subsets that would be created by adding the feature to the visualization. This means that the ratings take into account features that are already selected. For example, suppose that we have a dataset about people with two discrete features: age (young, middle, or old) and height (short or tall). If there are no selected features, then the rating for the age feature is calculated based on the three data subsets created by the feature's bins. Likewise, the rating for the height feature is calculated based on the two data subsets created by the feature's bins. If we select the age feature and add it to the visualization, then the rating for the height feature is recalculated based on the six data subsets created by the intersections of the bins for age and height. The rating of each feature is updated when a feature is selected, removed, or edited, when the dataset is filtered, or when the rating metric is changed.

In the left sidebar of the user interface, the ratings are visualized with gray bars behind the feature names (Figure 1A). The length of a bar encodes the feature's rating. Regardless of the metric chosen, the feature ratings are normalized to be between 0 and 1. Features that are already selected are not assigned a rating.

4.3 Feature Combination Suggestions

The individual feature ratings can tell the user which feature they can add to their current selection in order to maximize (or minimize) their chosen metric. They provide incremental guidance to the user in order to support them as they explore the dataset. However, since the user is adding features one-by-one, these ratings do not tell the user from the start what combinations of features maximize the metric. To address this limitation, the second form of guidance that SliceLens offers is suggesting combinations of one to three features that rank highly according to the chosen metric.

We decided to not suggest combinations of four features for several reasons. First, the more features that are in a combination, the more difficult it is to reason about the relationship between those features. For example, Halford et al. found a significant decrease in performance when participants were processing the interactions between four variables when compared to their performance for three variables [7]. The visualization also becomes more complex with more features. For example, with four features selected, in the worst default case, each feature could have five bins, resulting in $5^4 = 625$ subsets. We wanted to avoid recommending feature combinations that may result in visualizations that are excessively difficult for the user to read. In addition, not considering feature combinations with more than three features simplifies the algorithmic problem and enables us to evaluate more combinations of one to three features in a given amount of time than we could if we also had to evaluate larger combinations.

The algorithm to generate these feature combinations first sorts the features in the dataset according to the chosen metric and takes the top K, where K is set by the user. These K individual features are added to the list of suggested combinations. The algorithm then calculates the rating for all pairs of the top K features. In order for a pair to be added to the list of suggested combinations, it must have a rating that is higher than the median rating of the top K features. The rationale for this is that adding an additional feature makes the visualization more complex, therefore it must offer an improvement in rating to be included. Following this, the algorithm then calculates the rating for all triplets of the top K features. A triplet is added to the list of suggested combinations if its rating is higher than the median rating of the pairs in the suggested combinations list. Finally, the algorithm sorts the list of suggested feature combinations by rating.

In the left sidebar of the user interface, the user can generate the suggestions via the "Suggest Combos" button (Figure 1A). The user can then click the arrow buttons or use the arrow keys on their keyboard to cycle through the visualizations of the suggested feature combinations. This lets the user quickly explore combinations of features without having to manually add and remove each feature. However, at any point, the user can still add, remove, or edit features in the suggested combinations.

5 LIMITATIONS AND FUTURE WORK

One limitation of SliceLens is the slow performance of the feature combination suggestion algorithm as the number of instances and features in the dataset grows. The current approach to run this algorithm in reasonable time for datasets with many features is to not consider feature combinations that contain features that rank outside of the top K for the chosen metric. This is not ideal, because these lowly ranked features may combine with other features to create high rating subsets, but such combinations are not found by the current algorithm. We are interested in receiving feedback from the HILDA community on how to improve the performance of the algorithm without having to exclude features.

An additional limitation of SliceLens is its handling of features that have heavily skewed distributions. The default equal-width bins can result in some subsets containing many instances and other subsets containing few. The equal-frequency bins can result in better splits in these cases, but they might still be ineffective if one value of the feature appears much more often than others. Therefore, the user may have to determine effective custom bin thresholds, which takes time away from their exploration of the data. One possible improvement is that SliceLens could suggest bin thresholds or provide smarter defaults, possibly based on the chosen rating metric. For example, the tool could suggest thresholds that would lead to the purest bins for the feature.

While binning quantitative features is a good approach to evaluate model performance across different data subsets, it is not the best approach to understand the relationship between the dataset's features and labels, since information is lost when discretizing quantitative features. In future work, we are interested in focusing on guiding users in exploring machine learning datasets before they train a model, without also supporting exploring the performance of the model across different subsets after training. This would enable us to more effectively guide users in understanding the feature-label relationships in datasets that only have ground truth labels, as we would not need to do subset analysis and split quantitative features into bins. This would let us use more standard visualizations, such as scatter plots.

REFERENCES

[1] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17, 12, (Dec. 2011), 2301–2309. DOI: 10.1109/TVCG.2011.185.

[2] Á. A. Cabrera, W. Epperson, F. Hohman, M. Kahng, J. Morgenstern, and D. H. Chau. 2019. Fairvis: visual analytics for discovering intersectional bias in machine learning. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 46–56.

[3] William S. Cleveland and Robert McGill. 1984. Graphical perception: theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79, 387, 531–554. http://www.jstor.org/stable/2288400.

[4] Dheeru Dua and Casey Graff. 2017. UCI machine learning repository. (2017). http://archive.ics.uci.edu/ml.

[5] [SW], Facets 2017. URL: https://pair-code.github.io/facets/, VCS: https://github.com/PAIR-code/facets.

[6] Hadi Fanaee-T and Joao Gama. 2013. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 1–15. DOI: 10.1007/s13748-013-0040-3.

[7] Graeme S. Halford, Rosemary Baker, Julie E. McCredden, and John D. Bain. 2005. How many variables can humans process? *Psychological Science*, 16, 1, 70–76. PMID: 15660854. eprint: https://doi.org/10.1111/j.0956-7976.2005.00782.x. DOI: 10.1111/j.0956-7976.2005.00782.x.

[8] [SW] Rich Harris, Svelte 2016. URL: https://svelte.dev, VCS: https://github.com/sveltejs/svelte.

[9] Minsuk Kahng, Dezhi Fang, and Duen Horng (Polo) Chau. 2016. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics* (HILDA '16) Article 1. Association for Computing Machinery, San Francisco, California, 6 pages. ISBN: 9781450342070. DOI: 10.1145/2939502.2939503.

[10] David Munechika, Zijie J. Wang, Jack Reidy, Josh Rubin, Krishna Gade, Krishnaram Kenthapadi, and Duen Horng Chau. 2022. Visual Auditor: Interactive Visualization for Detection and Summarization of Model Biases. In *2022 IEEE Visualization and Visual Analytics (VIS)*. (Oct. 2022), 45–49. DOI: 10.1109/VIS54862.2022.00018.

[11] Neoklis Polyzotis, Steven Whang, Tim Klas Kraska, and Yeounoh Chung. 2019. Slice finder: automated data slicing for model validation. In *Proceedings of the IEEE Int' Conf. on Data Engineering (ICDE), 2019*. https://arxiv.org/pdf/1807.06068.pdf.

[12] J. R. Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1, 1, 81–106. ISBN: 1573-0565. DOI: 10.1007/BF00116251.

[13] Donghao Ren, Saleema Amershi, Bongshin Lee, Jina Suh, and Jason D. Williams. 2017. Squares: supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 23, 1, 61–70. DOI: 10.1109/TVCG.2016.2598828.

[14] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2020. The what-if tool: interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26, 1, 56–65. DOI: 10.1109/TVCG.2019.2934619.

[15] Xiaoyu Zhang, Jorge Piazentin Ono, Huan Song, Liang Gou, Kwan-Liu Ma, and Liu Ren. 2022. SliceTeller : A Data Slice-Driven Approach for Machine Learning Model Validation. *IEEE Transactions on Visualization and Computer Graphics*, 1–11. DOI: 10.1109/TVCG.2022.3209465.

Received 30 March 2023