SUBPLEX: A Visual Analytics Approach to Understand Local Model Explanations at the Subpopulation Level

Jun Yuan and Gromit Yeuk-Yin Chan, *New York University, Brooklyn, NY, 11201, USA*

Brian Barr, Kyle Overton, and Kim Rees, *Capital One, McLean, VA, 22102, USA*

Luis Gustavo Nonato, *Universidade de São Paulo, São Paulo, 13560-970, Brazil*

Enrico Bertini, *Northeastern University, Boston, MA, 02115, USA*

Claudio T. Silva, *New York University, Brooklyn, NY, 11201, USA*

Understanding the interpretation of machine learning (ML) models has been of paramount importance when making decisions with societal impacts, such as transport control, financial activities, and medical diagnosis. While local explanation techniques are popular methods to interpret ML models on a single instance, they do not scale to the understanding of a model's behavior on the whole dataset. In this article, we outline the challenges and needs of visually analyzing local explanations and propose SUBPLEX, *a visual analytics approach to help users understand local explanations with subpopulation visual analysis.* SUBPLEX *provides steerable clustering and projection visualization techniques that allow users to derive interpretable subpopulations of local explanations with users' expertise. We evaluate our approach through two use cases and experts' feedback.*

Machine learning (ML) produces accurate prediction models that can be applied to address important societal problems. While the advance of ML attracts a broader application in different domains, its increasing influence also raises a greater demand for transparency and interpretation of the decisions an ML model makes. Thus, the ML community has devoted efforts to explaining an ML model's behavior, regarding or regardless of a model's internal structure. A popular approach is to generate a *local explanation* per input to a model[1,2] where the explanation is a feature vector that explains the importance of each feature on an input to the model's decision. These methods are popular due to accurate results and easy-to-understand representations for end users.

However, one challenge of using local explanation is the lack of a *global view* on an ML model. By focusing on explaining a single instance out of all instances in the whole dataset, users might lose context on how the model behaves on a much larger population. Therefore, to understand a large number of local explanations, it is worthwhile to apply visual analysis[3] since users can acquire a big picture of the ML model by exploring all local explanations of a dataset at once through their visual representations. To be specific, users can visually group similar local explanations to gain an overview of different methods the model uses to make decisions. Then, they can understand each way of explanation by exploring each group of local explanations. We call such a process "visual analysis on local explanation subpopulations."

To conduct such visual analysis, users can apply high-dimensional data visualization techniques with well-studied methods, such as clustering and projections.[4] However, we find out that a straightforward solution of using clustering and projection on local explanations does not work well for the following reasons. First, as we will demonstrate in the following sections, the data characteristics of local explanations often hinder the effectiveness of clustering and projection results from revealing an accurate overview of a model's behavior. Second, through working with domain experts, we discovered that users need a

human-in-the-loop mechanism to steer the local explanation explorations and identify important features among all the features generated in a local explanation. Third, users need the system to be embedded in their data science workflow.

Having these challenges in mind, we designed *SUBPLEX*, a visual analytics approach that visualizes ML model local explanations at a subpopulation level. It is designed through an iterative discussion with a team of data scientists. We combine the concepts of subpopulation analysis in visualization and real industrial tasks on model interpretability for model developers to induce a feedback-driven analysis of subpopulation analysis on local explanations at scale. In short, our contributions include the following.

1) A human-in-the-loop framework for local explanation subpopulation analysis.
2) An interactive visual analytics tool embedded in Jupyter notebook, *SUBPLEX*, based on an iterative design process.
3) Two use cases and user feedback collection as evaluation of *SUBPLEX*.

BACKGROUND AND RELATED WORK

Local Explanations for Model Understanding

To deal with the inherent complexity of ML models and their lack of transparency, a number of "post-hoc" explanations or "attribution" methods have been developed. In this work, we focus on the local explanations, which help humans understand how individual prediction is made.

Current local explanations usually express the *explanation* for a data point as a vector where each value inside the vector is a human-understandable object. For example, additive feature explanation methods, such as LIME,[2] SHAP,[5] and GAM[6] output the explanations as a list of feature importance. Therefore, we can define an explanation a for an input as a real-valued explanation vector mapped to a feature space with m features

$$a = \{w_1, w_2, \ldots, w_m\} \forall w_i \in \mathbb{R} \qquad (1)$$

where each weight w_i represents the importance of a feature W_i.

In this work, we focus exclusively on local explanations in the format of the attribution vector a. We aim to help data scientists to discover and interpret subpopulations with different local explanation patterns.

Visual Analytics of Local Explanations

To interpret local explanations on a set of instances, people calculate aggregated explanation values, for example, the average of the absolute SHAP values.[7] However, average values over the whole dataset can be misleading because a feature might only be "important" for a subgroup but not the whole dataset. Lundberg et al.[8] introduce the analysis of SHAP values in clusters. However, they only use a straightforward clustering approach. In our work, we support both automatic clustering and manual subpopulation creation. Recent visual analytics systems, such as DECE,[9] also help in understanding model behavior on a subpopulation of data using rules. Our work, instead, focuses on local explanation vectors.

Recently, more visual analytics techniques have been proposed to help users analyze local explanations. Collaris et al.[10] introduced local/global contribution-value plots by showing how a local explanation changes different given feature values. Similarly, ExplainExplore[11] helps users identify instances classified with similar reasons by plotting all the LIME[2] values. Our approach, instead, automatically identifies subpopulations with similar local explanations and then visualizes aggregated local explanations at the subpopulation level.

Dimension reduction algorithms and clustering algorithms are both frequently used techniques in visual analytics for subpopulation-level data analysis. Recent surveys[4,12] list how they are combined in practice. In general, clustering implies visually highlighting similar instances (e.g., with the same color), while projections imply visualizing the samples as dots in a scatter plot with positions correlated to the similarities among the instances. We apply the two methods in our work with steerable mechanisms focusing on the characteristics of local explanations to aid the analysis.

SUBPOPULATION ANALYSIS ON LOCAL EXPLANATIONS

In this section, we introduce the technical challenges of visually analyzing local explanations and propose a steerable subpopulation analysis pipeline to address the technical challenges.

Clustering and Projections

A subpopulation analysis implies that we are interested in grouping the local explanations across the dataset based on their similarities. Grouping allows us to aggregate the explanations and reduce the visual complexity to examine a large set of explanations
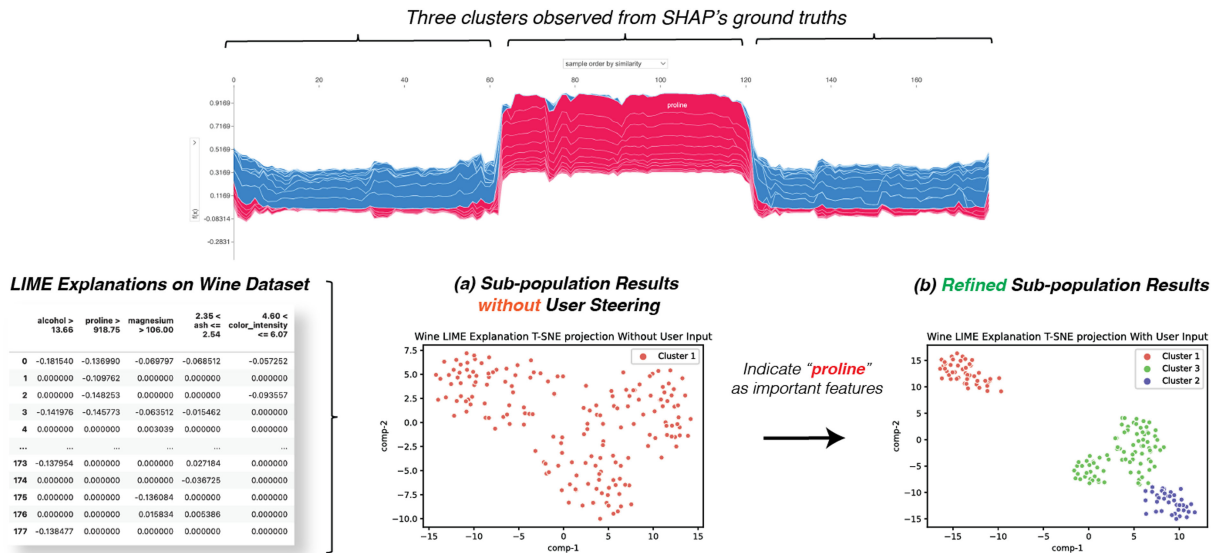
FIGURE 1. Illustration of refining LIME subpopulation results with a synthetic dataset. The ground truth of the classifier from SHAP contains *three* decision boundaries. (a) Original subpopulation result of the explanations does not reflect the ground truth. (b) When selecting the features related to *proline,* which are the main rationale of the model, the subpopulation result reflects three groups and becomes consistent with the ground truth.

simultaneously. In data visualization, the popular approaches are clustering and projections[4] as we describe in the previous section.

While we might address our goal of analyzing local explanations at scale by directly applying these methods, our collaborations with domain experts revealed several technical challenges. These challenges required us to provide additional metrics and mechanisms to facilitate the visual analytics process. Before introducing the system design requirements in the next section, our technical challenges from the data (i.e., local explanations) perspective are as follows:

1) *Local explanations are sparse and noisy:* The comparison between local explanations as a whole is not meaningful since some features are more important than others due to sparsity and noisiness in the explanation outputs. The distance metric for the subpopulation result should not treat features equally.

2) *Many features exist in the explanations to inspect or refine:* To identify important features to group explanations or discover the important features, users need to inspect many features. Without feature ranking or filtering, there will be a lot of human effort in the analysis.

To illustrate how the abovementioned shortcomings will affect the subpopulation results of local

explanation, we present a result using a synthetic wine classification dataset[13] (see Figure 1). To obtain a ground truth on how many groups of rationale exist in the dataset and classifier, we train a random forest classifier and obtain an SHAP explanation[5] on the data. We choose the random forest classifier as the illustration since the SHAP algorithm can generate accurate explanations and the number of subpopulation clusters on tree-based classifiers. Upon knowing there are three clusters of explanations from SHAP's plot (Figure 1), we generate other local explanations using LIME. The local explanations are harder to explore since they have 52 dimensions, and the values are sparse in general. We visualize the explanations with projections and colors generated from the DBSCAN clustering algorithm to highlight the number of similar groups in the explanations. The results are shown in Figure 1(a). We cannot observe any significant group structure from either the clusters or projections due to the high dimensionality and sparsity from the LIME explanations.

Steering Subpopulation Results

Seeing the abovementioned shortcomings of subpopulation analysis for local explanations, we are motivated to introduce human input to refine the results of the computed subpopulations. Note that any pipeline of subpopulation techniques can be generalized by first constructing a distance measure among

instances (i.e., local explanations) and then applying a subpopulation technique (e.g., MDS, tSNE, and K-means clustering). Thus, to generalize our approach for choosing projection and clustering techniques with flexibility, we propose a steerable technique that focuses on adjusting the distance measure with human input.

Since most of the clustering and projection techniques take Euclidean distance between high-dimensional vectors as inputs, we define a user input-aware distance metric

$$d(a_i, a_j) = \sqrt{\sum_{k \in f_u} \alpha (w_{ik} - w_{j_k})^2 + \sum_{k \in f_n} \beta (w_{ik} - w_{j_k})^2} \quad (2)$$

where f_u is the features in the explanations that the user thinks it is important, and f_n is the remaining features. α and β are the constants that we set up to adjust the influences of the features to reflect the user's emphasis. By default, $\alpha = 0.9$ and $\beta = 0.1$.

To understand how it improves the result, we use the same synthetic dataset in Figure 1. If we add the features that reflect the decision boundaries (i.e., "proline") to f_u, we can see the grouping becomes more related to the ground truths [see Figure 1(b)], where there exist three groups from the clustering and projection results.

Recommending Features for Steering
Since feature space might be huge, to assist users in refining the clustering results, we present feature suggestions and enable feature customization for cluster generation.

We now define each feature across the whole explanation dataset as W_i, and our goal here is to provide a score for each feature [i.e., $\mathrm{score}(W_i)$]. When a clustering result is generated, we obtain a set of clusters $\mathcal{C} = \{C_1, C_2, \ldots\}$, where C represents a set of similar explanations. Inspired by cluster evaluation methods,[14] we propose three properties to score features:

Property 1 (Consistency): The variance of W_i within each cluster

$$\mathrm{cst}(W_i) = \sum_{C \in \mathcal{C}} \|C\| / \mathrm{var}(\{w_{j_i} | a_j \in C\}) \quad (3)$$

where $\|C\|$ is the number of instances in the cluster C. The higher the $\mathrm{cst}(W_i)$, the more consistent the local explanations of feature W_i are distributed in each cluster.

Property 2 (Separation): The difference of values of W_i across different clusters

$$\mathrm{sep}(W_i) = \mathrm{var}(\{\mathrm{mean}(\{w_{j_i} | a_j \in C\}) | C \in \mathcal{C}\}). \quad (4)$$

The higher the $\mathrm{sep}(W_i)$ is, the more distinctive the local explanations of feature W_i are in different clusters.

Property 3 (Importance): The significance of W_i in one or more clusters

$$\mathrm{impt}(W_i) = \max(\mathrm{mean}(\{w_{j_i} | a_j \in C\})). \quad (5)$$

The higher the $\mathrm{impt}(W_i)$ is, the more important feature is in at least one cluster.

Under Properties 1–3, for a given feature W_i and the present clusters, we calculate a feature score

$$\mathrm{score}(W_i) = \overline{\mathrm{cst}(W_i)} + \overline{\mathrm{sep}(W_i)} + \overline{\mathrm{impt}(W_i)} \quad (6)$$

where $\overline{(\cdot)}$ normalizes the values within the range of 0–1.

Once the users run the clustering algorithm, we suggest the top *k* features with the highest feature scores (default $k = 6$).

HUMAN-IN-THE-LOOP PIPELINE
In this section, we introduce the design rationale of *SUBPLEX*. *SUBPLEX* is developed as a part of an explainable artificial intelligence project. We held biweekly meetings with experts in ML and visualization for 12 months, at which a variety of design requirements were gathered. Together with the computation and steering mechanism in the previous section, we developed the visual analytics system via the iterative user-centered design process. In the following, we first connect the design goal with specific analytical tasks through hierarchical task analysis[17] and task abstraction.[16] Then, we list the design requirements and provide an overview of the resulting human-in-the-loop pipeline.

Hierarchical Task Abstraction (HTA) and Design Requirements
The goal of this work is to help data scientists understand model logic through local model explanations. More specifically, we want to help answer the question of "How do different features influence the model predictions?"

To better support the goal of local explanation analysis, we conduct hierarchical task analysis (see Figure 2),[17,18] to understand the workflow. Then, we apply the framework proposed by Lam et al.[16] to
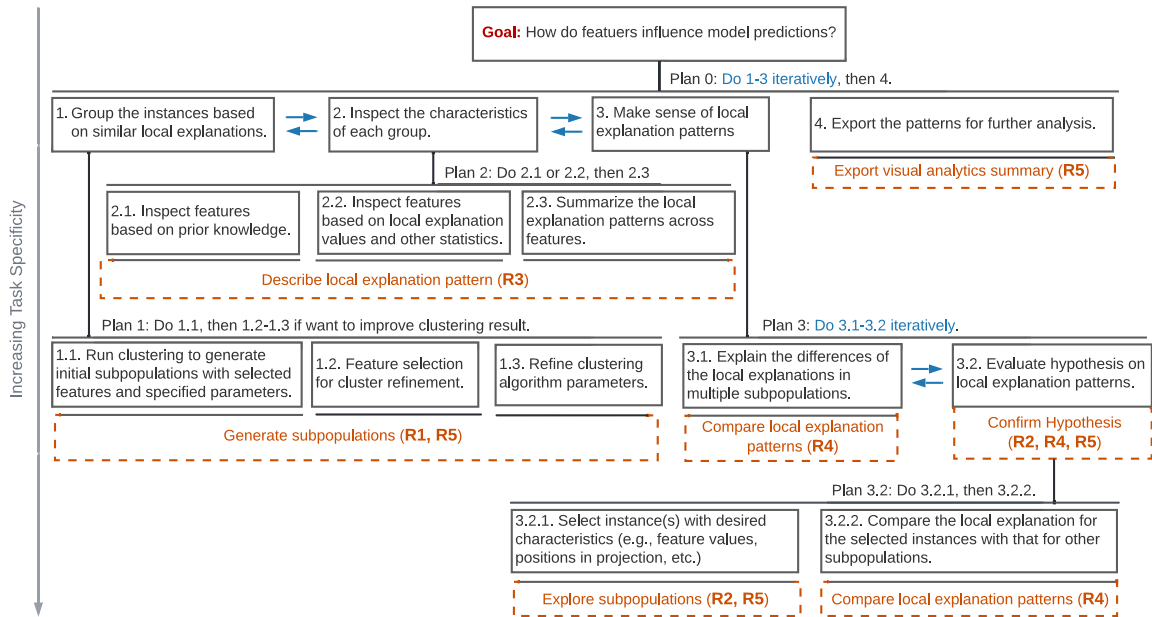
FIGURE 2. HTA of local explanation analysis using box-and-line notation. We follow the standard conventions for hierarchical task analysis[15] where tasks are represented by named boxes with a unique ID, which also indicates the hierarchical level of the task. Task abstraction based on Lam et al.'s[16] work are highlighted in orange. The related design requirements are denoted with each abstract task.
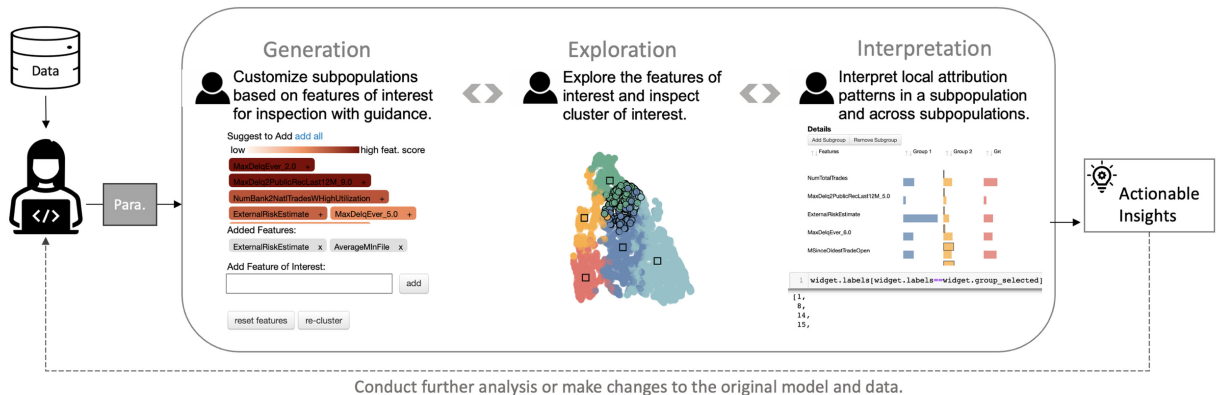

FIGURE 3. Overview of the human-in-the-loop pipeline for local explanation analysis in the coding environment.

connect the analytical goal with abstract tasks to derive the design requirements. The four requirements (R1-4) from HTA with a general data science workflow requirement (R5) are as follows:

R1: Guided feature selection for clustering refinement: To mitigate the effects of noisy features, we should guide users to identify features that lead to clear clusters.

R2: Flexible support to analyze user-defined subpopulations: Data scientists may have instances they are interested in based on prior knowledge. Therefore,

our tool should support flexible subpopulation creation by interactive selection and queries.

R3: Visual assistance for identifying local explanation pattern in one subpopulation: Our tool should support interactive exploration to identify patterns, including the size of a cluster, feature distribution in a subpopulation, and instances in a subpopulation.

R4: Visual comparison of local explanation patterns in different subpopulations: Our tool should support visual comparison of local explanation patterns for interpretation and validation of the local explanations.
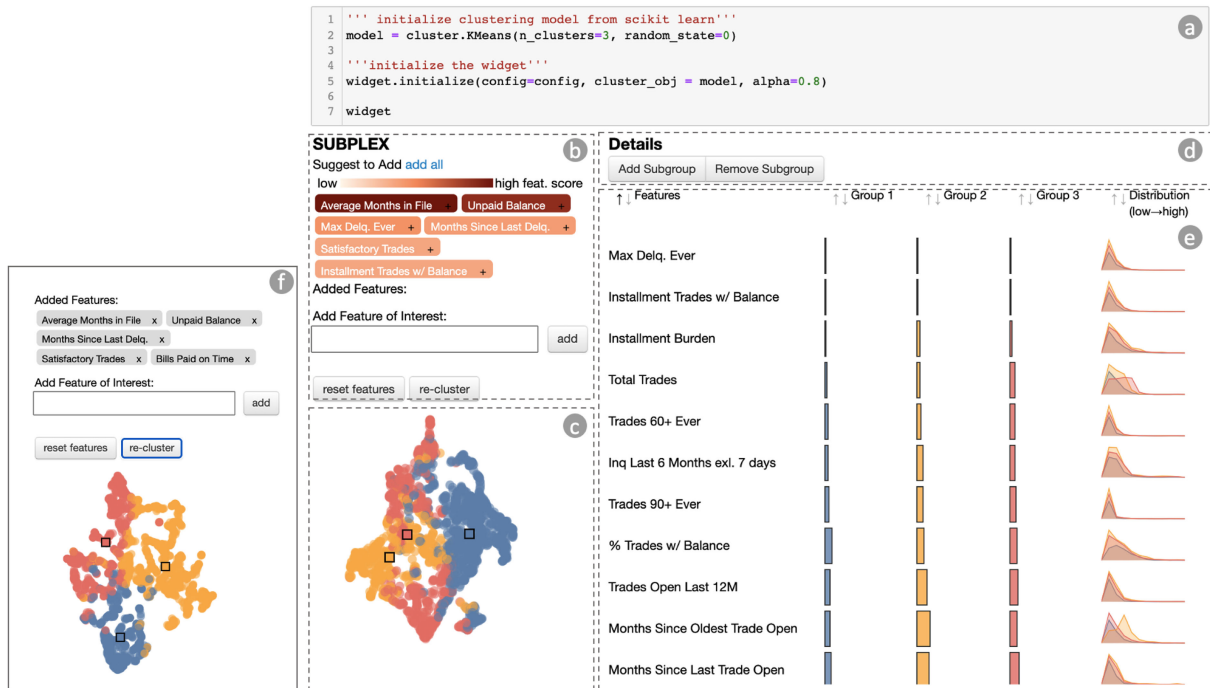
```
1  ''' initialize clustering model from scikit learn'''
2  model = cluster.KMeans(n_clusters=3, random_state=0)
3
4  '''initialize the widget'''
5  widget.initialize(config=config, cluster_obj = model, alpha=0.8)
6
7  widget
```

FIGURE 4. *SUBPLEX* contains five linked views: (a) code block, (b) cluster refinement view, (c) projection view, (d) subpopulation creation panel, (e) local explanation detail view.

R5: Coding-environment-friendly visual analytics: A general data science system requirement is to embed the system in computation notebooks to take the processed data for visualization and output results for later model development and deployment.

System Overview

Based on the HTA, we introduce a human-in-the-loop approach for subpopulation-level local explanation analysis. It contains three main stages, as shown in Figure 3.

Generation: The first stage focuses on the automatic subpopulation generation (clustering) and refinement (R1). After, the widget is initialized with the explanations to be analyzed, we visualize the clusters in the 2-D space. We then suggest the top features with the highest feature scores.

Exploration: The second stage is to explore a specific subpopulation (R2, R3). Users can manually add/remove subpopulations (R2) by multiple means and identify the local explanation patterns (R3) in different subpopulations.

Interpretation: The third stage enables users to export intermediate results for further analysis in the notebook (R5) and test the hypothesis they have in mind by comparing the explanation patterns (R4) in manually created subpopulations (R2).

To instantiate this pipeline, we developed *SUBPLEX*, an interactive widget embedded in the Jupyter notebook. In this way, data scientists can interactively analyze local explanations using code and conventional interactions, such as clicking and brushing. Moreover, the intermediate result of the visual analysis can be exported as a variable (e.g., DataFrame and list in Python) for further analysis.

SUBPLEX: VISUALIZATION AND INTERACTION

As shown in Figure 4, the *SUBPLEX* includes five views to assist users to understand and analyze local explanations:

a) *the code block* inside a Jupyter notebook enables users to set user-defined parameters and export analysis results as variables;
b) *the cluster refinement view* provides feature suggestions and allows users to refine clusters;
c) *the projection view* shows the distribution of local explanations;
d) *the subpopulation creation panel* enables manual add-and-remove for subpopulations;

```
1  indices = df[(df['ExternalRiskEstimate']>90)&(df['label']==0)].index
2  widget.set_highlight_instance(indices.tolist())
```
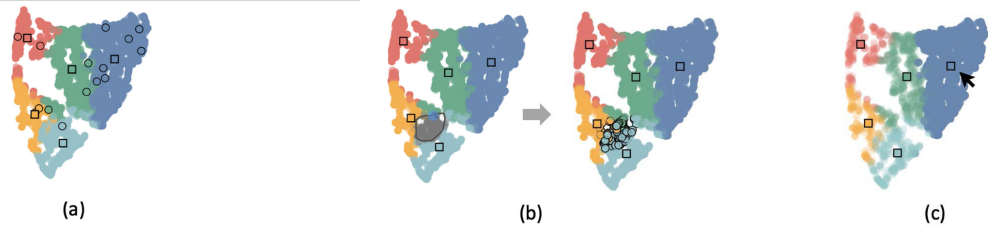

FIGURE 5. Three methods of selecting/creating a subpopulation for inspection. (a) Highlight instances by coding. (b) Select instances by brushing. (c) Highlight a cluster by clicking centroid.

e) *the local explanation detail view* displays the aggregated local explanation distribution for each subpopulation.

In the following sections, we introduce the coordinated views and interactions according to how they support the three stages in the pipeline and fulfill the design requirements.

Generation

The widget is initialized with the feature names, an array of local explanations, and the automatic clustering model users prefer to use. As shown in Figure 4(a), the user inputs the feature names and local explanations in `config`, feed in the widget a predefined `cluster_obj`, and specify `alpha` (α) for later cluster refinement. Then, the widget renders below the code block (R5).

In the *cluster refinement view* [see Figure 4(b)], we display the top six features with the highest feature scores (R1). Users can directly add the recommended features by clicking "+" or typing the feature names to refine clusters.

Exploration

The *projection view* [see Figure 4(c)] presents the local explanation distribution in a 2-D space. Based on the weighted local explanations, we generate a 2-D position for each local explanation vector using UMAP.[19] With the distribution of local explanations, users can gain a general idea of how the clusters distribute and how large each cluster is (R3).

In the *local explanation detail view* [see Figure 4 (e)], each row shows the explanations for one feature. The first column is always the feature name, which follows the bars representing the average local explanation values in different subpopulations. The last column of each row displays the histogram of the local explanation values in different subpopulations. We enable feature sorting based on explanation values in

one or more subpopulations to facilitate pattern identifications (R3). Users can click the "Distribution" column to sort the features based on the local explanation variances among the clusters. In both the *projection view* and the *local explanation detail view*, we use the same categorical color scheme to encode different subpopulations.

As shown in Figure 5, users can define subpopulations in three ways (R2). The first method is specifying the instances of interest. By running `set_highlight_instance()` in the *code block*, the user-defined instances will be highlighted in the projection. The second method is brushing instances in the *projection view*. The third method is clicking the centroid points (squares in the projection) to highlight all the instances in the selected cluster. These selection methods allow users to quickly acquire their desired set of explanations for analysis. By default, any selected dots are highlighted with a black edge as shown in Figure 5(a) and (b). However, if users focus on a specific cluster that usually contains a lot of points, showing all the selected points with black edges will cause visual clutter. Thus, we fade the dots in unselected clusters to highlight the selected cluster, as shown in Figure 5(c).

Interpretation

We enable visual comparison of local explanation patterns in multiple subpopulations in the *local explanation detail view*. In this view, the juxtaposition of bars enables visual comparison of the aggregated local explanations in different groups (R4).

To support the comparison with user-defined subpopulations (R2), the tool automatically calculates the average local explanation of a user-defined group. As shown in Figure 8(b), the average value of the selected instances are shown as rectangles with black edges, while the average explanation value for an existing group/cluster is shown as a rectangle without black edges. After clicking the *Add Subgroup* button
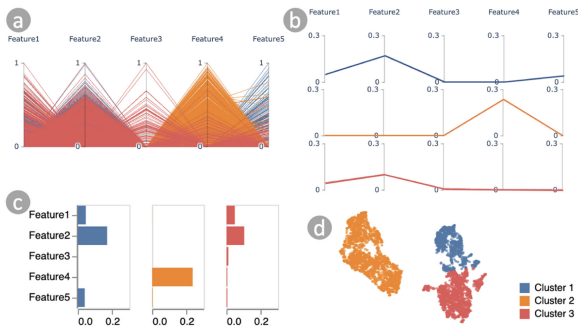
FIGURE 6. Alternative designs for subpopulation analysis. (a) Parallel coordinates plot for all the explanations. (b) Parallel coordinates plot for aggregated explanations. (c) Bar charts for aggregated explanations. (d) Projection of explanations in a scatter plot.

[see Figure 4(d)], a new column of bars for this subgroup will be added. To confirm the hypotheses about the relationship between feature values and the local explanations, users can export the feature value histogram by calling `plot_val_histogram()` [see Figure 7(b)], and check the local explanations for the feature with specific values by calling `plot_customized_expl()` [see Figure 7(c)].

SUBPLEX enables users to export the analysis results and store them as a variable in the Jupyter notebook (R5) by compiling code in the *code block,* as shown in Figure 8(c)–(e).

Several design alternatives for the subpopulation analysis were considered (see Figure 6). The first one was to use a parallel coordinates plot [see Figure 6(a)]. Although it can show all the local explanations in a subpopulation, this plot easily becomes cluttered, which makes it hard to interpret the pattern in a subpopulation. To help with pattern identification (R3), we can visualize aggregated local explanations (e.g., mean values) as shown in Figure 6(b). Compared with parallel coordinates, aligned bar charts [Figure 6(c)] have stronger color stimulus and are conventional. Therefore, we use bar charts to visualize aggregated local attributions. Moreover, compared with these charts providing specific explanation values, a projection of local explanations [see Figure 6(d)] can better demonstrate the spatial relationship and distribution of multiple subpopulations. As a result, we include the *projection view* in *SUBPLEX* with various interactions to support visual analysis.

EVALUATION

To demonstrate how *SUBPLEX* facilitates local explanation analysis, we first present a usage scenario of
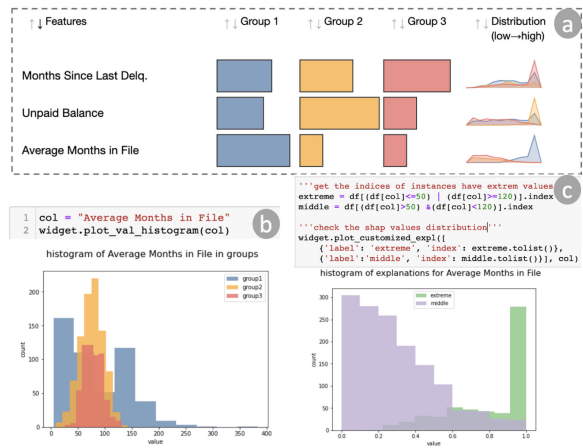

FIGURE 7. Analysis of local explanation patterns for a loan application dataset. (a) Top three features with the highest aggregated local explanation values. (b) Histogram plot of the local explanations in three subgroups in terms of a user defined feature. (c) Histogram plot for hypothesis testing.

analyzing local explanations for a loan application model. Then, we demonstrate how the steerable clustering and projection techniques enable local explanation analysis for high-dimensional data, such as text. In the end, we describe the feedback on *SUBPLEX* from eight domain experts.

Usage Scenario: Loan Application

A model developer, Alice, trains an SVM model on a set of loan applications.[a] This model predicts whether a loan applicant will default or not with a test accuracy of 73.6%. Before the model deployment, Alice wants to understand how the model behaves. Thus, Alice generates SHAP values for all the test instances for analysis, which takes 0.37 s.

Cluster generation and refinement: After running the widget, the initial results of the automatically generated clusters are not well separated in the projection view [see Figure 4(c)] due to many features (i.e., 22) in the explanation vectors. Thus she browses the feature suggestions [see Figure 4(b)] and adds three features *Average Months in File*, *Unpaid Balance*, and *Months since Last Delq.* and manually selects *Satisfactory Trades* and *Bills Paid on Time* (R1). Then she regenerates the clusters and gets more separated results [see Figure 4(f)]. Throughout the

[a][Online]. Available: https://community.fico.com/s/explainable-machine-learning-challenge
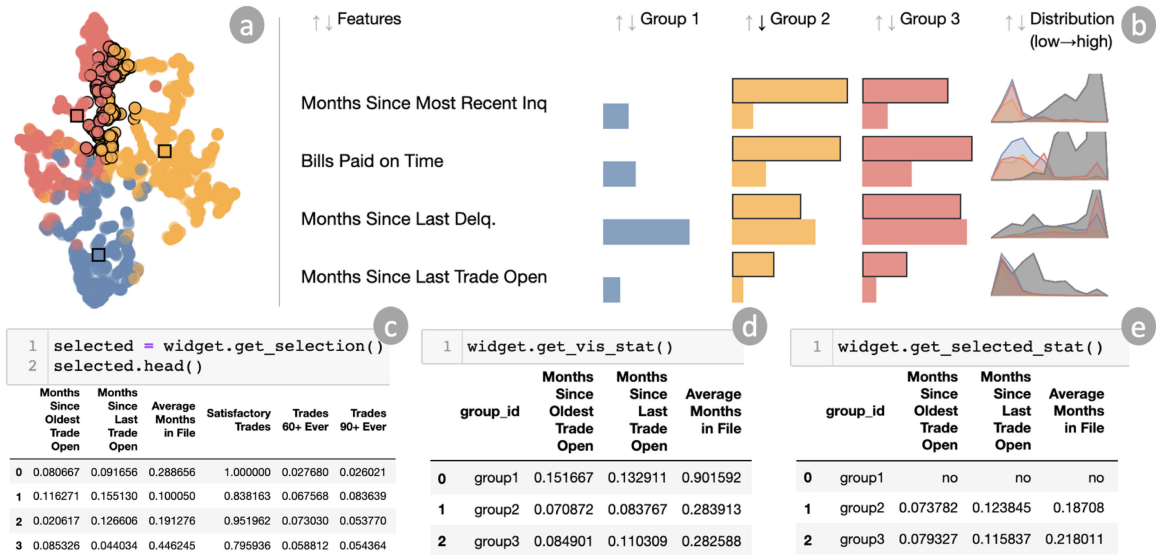
FIGURE 8. Brush selection and then output the aggregated information as a DataFrame in Python. (a) Select data points at the decision boundary. (b) Visualize local explanation values for the selected data points. (c)–(e) Output the statistics about the selected data points to a DataFrame.

analysis, the widget takes 5–8 s to generate clusters and calculate the projection for 1975 instances.

Pattern identification: By looking at the table [see Figure 7(a)], she finds some patterns (R3): *Months Since Last Delq.* is important among all the three clusters. However, *Unpaid Balance* has a strong influence in Group 2. Also, *Average Months in File* has a strong influence in Group 1. Thus, Alice wants to understand why these features provide different ML models' rationale for the dataset.

Confirm hypothesis. Alice first exports the histogram (R5) for the *actual values* of *Average Months in File* in the three groups. For group 1, the values are between 0 and 400, with two peaks around 50 and 120 [blue bars in Figure 7(b)]. Yet, groups 2 and 3 have small values, mostly between 50 and 120. Alice now has a hypothesis that the feature *Average Months in File* has a stronger influence when it has extreme values such as fewer than 50 or more than 120. To verify, Alice defines two groups of extreme values (either < 50 or >120) and middle values (between 50 and 120) for *Average Months in File* (R2, R5). Then she runs a query, and the widget outputs the histogram of the local explanation values for the two defined groups (R4, R5). As shown by the green bars in Figure 7(c), the extreme values of *Average Months in File* have strong influences on the model predictions.

Pattern identification: Alice wants to understand the local explanations around the group boundaries for further analysis. So she brushes the projection and selects boundary instances [see Figure 8(a)]. In the Detail view [see Figure 8(b)], Alice observes how the local explanations of boundary instances are different from the discovered groups. It shows that the boundary instances are influenced much more by *Months Since Most Recent Inquiry* and *Bills Paid On Time* compared with other instances outside the boundary.

Summary export: After learning the model's behavior, Alice exports the summary visualization and the aggregated local explanation from different groups as a Python DataFrame [Figure 8(c)–(e)] for later discussion with her colleagues (R5).

Usage Scenario: Sentiment Analysis on Tweets

A data scientist, Bob, is considering using an opensourced model *twitter-roberta-base-sentiment*[20] to predict whether a tweet contains a *positive*, *negative*, or *neutral* sentiment. He runs the model on the test data[20] of 12,284 tweets and gets an overall accuracy of 72%. He then wants to gain a general idea of how the model makes predictions.

Cluster generation and refinement. Bob first generates and stores the SHAP values[5] of each instance for the class predicted by the model for analysis. It takes more than one hour to generate SHAP values for all instances with this large language model. For a text instance (e.g., a tweet), each token (word) is a feature. After an initial processing step of tokenization and
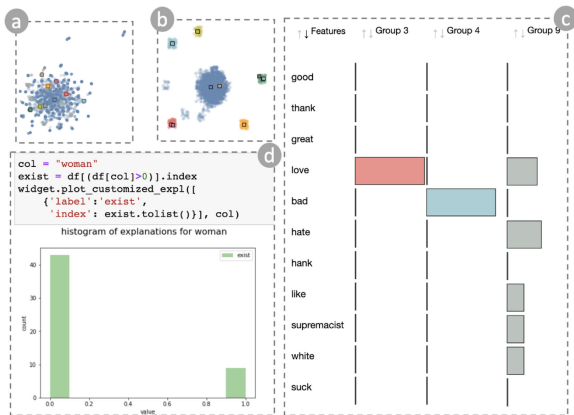
FIGURE 9. Usage scanario of understanding the sentiment analysis model of tweets. (a) Original clustering and projection of local explanations. (b) Refined clustering and projection. (c) Local explanation patterns in subgroups. (d) Distribution of local explanation values in terms of the token "woman."

lemmatization, the SHAP values for the whole test set can be transformed into an array of 12,284 instances with 17,000 features. After removing features (tokens) with low SHAP values and stop words (i.e., "to," "in," etc.), Bob starts the local explanation analysis based on 137 features.

Bob initializes *SUBPLEX* and tries to generate 10 clusters at the beginning. Soon he realizes that the result does not contain a clear visual structure [see Figure 9 (a)]. Thus, he checks the feature suggestions and notices that the tokens "thank," "love," "good," and "bad" have high feature scores while providing intrinsic meanings to him for understanding the sentiment in texts. As a result, Bob selects these features and reruns the clustering [see Figure 9(b)]. The local explanations in the discovered clusters are well separated now. For the inspected data (137 features/tokens, 12,284 instances), it takes around 16.7–17.5 s for the widget to generate clusters and calculate the projection.

Pattern identification and interpretation: Bob then inspects the local explanation detail view to extract the local explanation patterns. For example, Groups 3 and 4 shown in Figure 9(c), are mostly influenced by a single feature/token (e.g.,"love," "bad"); some groups are influenced by the combination of similar tokens (e.g., "love" and "like"). As shown in Figure 9(c), besides verbs related to emotions, tweets in Group 9 are influenced by nonemotional words, such as "supremacist." Bob then outputs the actual tweets and finds the tweets in this group are mostly complaints and labeled as negative sentiments.

Hypothesis testing: During the inspection, Bob notices that the token "woman" has high SHAP values in some subgroups. So he has a hypothesis that the models learned a biased correlation between "woman" and the sentiment. To test this, Bob plots the SHAP value distribution of "woman" and its variants (e.g., "women") for tweets that mention them by calling `plot_customized_expl()`. As shown in Figure 9 (d), there are around 50 tweets containing "woman" in total, where "woman" is extremely important in 10 tweets. Although the sample size is too small to conclude that the model learned a bias, more inspection is needed for cases related to gender bias in the language model.

Evaluation Via Expert Feedback

We presented the widget to eight domain experts in a commercial bank and collected their feedback. They are all data scientists specializing in financial data and used the Jupyter notebook in their work. They all had experience with model development, visualization, and local explanations. Among them, E2 is one of the ML experts who participated in the whole design process and helped us identify the design requirements, while the rest had no exposure to *SUBPLEX* before.

Procedure: Interviews were conducted remotely and lasted around 30 minutes each. We began each interview with an introduction, during which we clarified the goal of *SUBPLEX* and provided a tutorial on the tool. Then, we presented the case study of loan application as described in the first usage scenario to the experts. They were allowed to ask any questions or give any comments. In the final phase, we conducted an interview that incorporated several questions about the overall usefulness, comparison with the tools they were using, and general pros and cons of *SUBPLEX*.

Usefulness: In general, the experts found *SUBPLEX* useful. They liked the visualization of the automatic recommendations and then drill down into subpopulations of interest. The scatter plot and color encoding were intuitive for them to understand subpopulations. E1 stated that grouping the local explanations instead of aggregating the overall feature importance score would be more accurate. E2 mentioned that *SUBPLEX* helped bridge the gap between global explanations (too high-level) and individual local explanations (too many details). E3 liked the cluster visualization and found the workflow of the subpopulation discovery useful. E6 particularly liked the visual design where everything is approachable and interactive within a notebook environment, making it easy to integrate

into other work. Similarly, E8 liked the interactivity most because it allowed more exploratory analysis of model development with interpretability.

Feature/instance of interest: The experts had different preferences of features and subpopulations in exploring the local explanations. E1 expressed interest in the boundary instances. They wanted to understand how these instances ended up in different clusters. E3 preferred to explore age-related features to ensure they were not biased on age. Similarly, E6 would like to inspect sensitive demographic populations regarding age, race, gender, etc. E8, however, focused more on the features that are nonmonotonic to the target.

Comparison with other tools: The tools or methods used by the experts were mostly static plots generated via code or exclusive web applications. Compared with other methods, all of the experts liked the interactivity of *SUBPLEX*, which made the exploration more powerful. Moreover, E4 mentioned that compared with accumulated local effectsplot and partial dependence plots, which focus on just one or two features in one chart, *SUBPLEX* provided a more global view of data and explanations. E5 said that, in their typical work, they only scan through output automatically generated across all features/segments, while *SUBPLEX* provided intelligent pre-election in terms of suggesting features or clusters.

Suggestions for improvement: To further improve *SUBPLEX*, the experts suggested we provide more details on the dimensionality reduction method. E1 commented that different projection methods could change the analysis outcome. E5 wanted to include more details about the cluster/group creation details in the output because this can be easier to reproduce the results when discussing with other teams. We also found the comparison of local explanation methods a good next step in this work. For example, E2 wanted to explore and interpret the local explanation of *Bills Paid on Time* because, in another project, this feature was one of the most important features to change for a counterfactual explanation.

DISCUSSION

Design Guidelines
Expert feedback on *SUBPLEX* was positive. They liked the interactivity and found the visualization intuitive for them. We attribute these comments to our design process, where we took the workflow and analysis goal into consideration through the HTA. We expect the presented HTA can be helpful for future model explanation analysis tools.

Our target users are data scientists who have experience with static plots generated via coding. Therefore, we adapted the visualizations such as bar charts and scatterplots, which are mostly used by data scientists in their daily work. We also noticed that data scientists had some questions about the local explanations and the methods for clustering and projection. To better support the analysis, designs should be more transparent about the metrics and models used in the tool.

Lessons Learned
We discuss the lessons learned with respect to the following aspects.

Interacting with data in a coding environment is powerful: Our participants appreciated the interactive exploration of local explanations in Jupyter notebook. 87.5% (7/8) of our participants preferred a widget like *SUBPLEX* in the coding environment. Only one participant did not have a strong preference. We expect more visual analytics designs for data scientists embedded in the coding environment so that the users do not need to switch between different platforms.

Importance of the human-in-the-loop pipeline with intelligent UI: We also confirm the importance of involving humans in the loop with the assistance of an intelligent UI for local explanation analysis. Participants liked the guided clustering with feature suggestions while enabling manual feature adding. Participants expressed quite different interests in the features and instances for inspection, which requires the UI to support flexible feature/instance selection. We expect to see more work integrating human and machine intelligence in local explanation analysis.

Limitations
The system currently has certain limitations. First, our tool was developed with domain experts from a financial company, which might not represent all use cases concerning local explanations.

Second, the current widget needs to calculate clustering and projection, which requires some computation overhead. In our use cases, the computations took less than 10 seconds for around 2000 rows/20 columns and less than 20 seconds for about 12,000 rows/100 columns. We chose the clustering (K-Means) and projection (UMAP) techniques that are not quadratic in time complexity, but other similar methods, such as hierarchical clustering or tSNE, might not scale well.

Finally, by only interviewing eight experts from the financial domain, we may be overgeneralizing our

results. For the next step, we are interested in a controlled study that more directly compares subpopulation-level model explanation analysis with different visual representations, such as parallel coordinates and bar charts, as well as showing, aggregated local explanations or individual local explanations in a subpopulation.

CONCLUSION

In this work, through an iterative design process with expert ML researchers and practitioners, we propose steerable clustering and projection techniques to identify and select important features for an ML model's interpretation. We also identified a list of tasks for explaining an ML model using local explanations, which led to the design of *SUBPLEX*, a novel visual analytics tool in the Jupyter notebook environment. We demonstrate the effectiveness of our approach with two use cases and collect feedback from domain experts.

REFERENCES

1. B. Kim et al., "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)," 2017, *arXiv:1711.11279*.
2. M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you? Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1135–1144.
3. B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *The Craft of Information Visualization*. Amsterdam, The Netherlands: Elsevier, 2003, pp. 364–371.
4. J. Wenskovitch, I. Crandell, N. Ramakrishnan, L. House, and C. North, "Towards a systematic combination of dimension reduction and clustering in visual analytics," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 1, pp. 131–141, Jan. 2018.
5. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 31st Int. Conf. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4768–4777.
6. T. J. Hastie, "Generalized additive models," in *Statistical Models in S*. Evanston, IL, USA: Routledge, 2017, pp. 249–307.
7. C. Molnar, "Interpretable machine learning," 2019. [Online]. Available: https://christophm.github.io/interpretable-ml-book/
8. S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," 2018, *arXiv:1802.03888*.
9. F. Cheng, Y. Ming, and H. Qu, "Dece: Decision explorer with counterfactual explanations for machine learning models," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 2, pp. 1438–1447, Feb. 2021.
10. D. Collaris and J. J. van Wijk, "Comparative evaluation of contribution-value plots for machine learning understanding," *J. Vis.*, vol. 25, no. 1, pp. 47–57, 2022.
11. D. Collaris and J. van Wijk, "Explainexplore: Visual exploration of machine learning explanations," in *Proc. IEEE Pacific Vis. Symp.*, 2020, pp. 26–35.
12. L. G. Nonato and M. Aupetit, "Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 8, pp. 2650–2673, Aug. 2019.
13. D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
14. M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Cluster validity methods: Part I," *ACM Sigmod Rec.*, vol. 31, no. 2, pp. 40–45, 2002.
15. S. Kurniawan, "Interaction design: Beyond human-computer interaction by Preece Sharp and Rogers, (2001), isbn 0471492787," in *Universal Access in the Information Society*. Berlin, Germany: Springer, 2004.
16. H. Lam, M. Tory, and T. Munzner, "Bridging from goals to tasks with design study analysis reports," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 1, pp. 435–445, Jan. 2018.
17. P. Salmon, D. Jenkins, N. Stanton, and G. Walker, "Hierarchical task analysis vs. cognitive work analysis: Comparison of theory, methodology and contribution to system design," *Theor. Issues Ergonom. Sci.*, vol. 11, no. 6, pp. 504–531, 2010.
18. Y. Zhang, K. Chanana, and C. Dunne, "IDMVis: Temporal event sequence visualization for type 1 diabetes treatment decision support," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 512–522, Jan. 2019.
19. L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2018, *arXiv:1802.03426*.
20. F. Barbieri, J. Camacho-Collados, L. Espinosa-Anke, and L. Neves, "TweetEval: Unified benchmark and comparative evaluation for tweet classification," in *Proc. Findings Assoc. Comput. Linguistics: EMNLP*, 2020, pp. 1644–1650.

JUN YUAN is currently a Ph.D. candidate in the Department of Computer Science and Engineering, New York University, Brooklyn, NY, 11201, USA. Her research interest lies in the intersection of data visualization and explainable AI. She is the corresponding author of this article. Contact her at junyuan@nyu.edu.

GROMIT YEUK-YIN CHAN is with New York University, Brooklyn, NY, 11201, USA. His research focuses on machine learning aspect of interactive visualization. Chan received his Ph.D. degree in computer science from New York University, Brooklyn, NY, USA. Contact him at gromit.chan@nyu.edu.

BRIAN BARR is an explainable AI researcher with Capital One, McLean, VA, 22102, USA. His research interests include deep learning on tabular data, and local attribution methods. Barr received his Ph.D. degree in mechanical engineering from Clarkson University, Potsdam, NY, USA. Contact him at brian.barr@capitalone.com.

KYLE OVERTON is a visual/UX designer with Capital One, McLean, VA, 22102, USA. Overton received his M.S. degree in human–computer interaction from Indiana University Bloomington, Bloomington, IN, USA. Contact him at ko@kyleoverton.com.

KIM REES is currently the head of Data Experience Design with Capital One, McLean, VA, 22102, USA. She established the Responsible AI program and leads the Data Privacy design team in Capital One. Contact her at kim.rees@capitalone.com.

LUIS GUSTAVO NONATO is a full professor with Universidade de São Paulo, São Paulo, 13560-970, Brazil. His research interests include visualization, visual analytics, machine learning, data science, and geometric computing. He is a member of IEEE. Contact him at gnonato@icmc.usp.br.

ENRICO BERTINI is an associate professor with Northeastern University, Boston, MA, 02115, USA. His research focuses on the development and evaluation of interactive visual interfaces to help people reason with data and machine learning models. Contact him at e.bertini@northeastern.edu.

CLAUDIO T. SILVA is an institute professor of computer science and engineering with the NYU Tandon School of Engineering, New York, NY, 11201, USA, and a professor of data science with the NYU Center for Data Science, New York. His research interests include visualization and graphics, geometric computing, data science, urban computing, and sports analytics. He is a fellow of IEEE. Contact him at csilva@nyu.edu.