A Taxonomy of Property Measures to Unify Active Learning and Human-centered Approaches to Data Labeling

JÜRGEN BERNARD, University of British Columbia, Canada
MARCO HUTTER, TU Darmstadt, Germany
MICHAEL SEDLMAIR, University of Stuttgart, Germany
MATTHIAS ZEPPELZAUER, St. Pölten University of Applied Sciences, Austria
TAMARA MUNZNER, University of British Columbia, Canada

Strategies for selecting the next data instance to label, in service of generating labeled data for machine learning, have been considered separately in the machine learning literature on active learning and in the visual analytics literature on human-centered approaches. We propose a unified design space for instance selection strategies to support detailed and fine-grained analysis covering both of these perspectives. We identify a concise set of 15 properties, namely measureable characteristics of datasets or of machine learning models applied to them, that cover most of the strategies in these literatures. To quantify these properties, we introduce Property Measures (PM) as fine-grained building blocks that can be used to formalize instance selection strategies. In addition, we present a taxonomy of PMs to support the description, evaluation, and generation of PMs across four dimensions: machine learning (ML) *Model Output*, *Instance Relations*, *Measure Functionality*, and *Measure Valence*. We also create computational infrastructure to support qualitative visual data analysis: a visual analytics explainer for PMs built around an implementation of PMs using cascades of eight atomic functions. It supports eight analysis tasks, covering the analysis of datasets and ML models using visual comparison within and between PMs and groups of PMs, and over time during the interactive labeling process. We iteratively refined the PM taxonomy, the explainer, and the task abstraction in parallel with each other during a two-year formative process, and show evidence of their utility through a summative evaluation with the same infrastructure. This research builds a formal baseline for the better understanding of the commonalities and differences of instance selection strategies, which can serve as the stepping stone for the synthesis of novel strategies in future work.

CCS Concepts: • **Human-centered computing → Visual analytics**; • **Theory of computation → Active learning;**

The reviewing of this article was managed by special issue associate editors Shixia Liu, Daniel Archambault, Tatiana von Landesberger, Remco ChangCagatay Turkay.
This work was supported by a postdoc fellowship of the German Academic Exchange Service (DAAD). This research was further funded by the Austrian Research Promotion Agency (FFG) project 866880 "Big Data Analytics"; the Lower Austrian Research and Education Company (NFB), projects FTI17-014 and FTI18-005; and the National Sciences and Engineering Research Council of Canada project RGPIN-2014-06309. Further, this work was partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) by Project-ID 251654672–TRR 161 (project A08).
Authors' addresses: J. Bernard and T. Munzner, University of British Columbia, 201-2366 Main Mall, Vancouver, V6T1Z4, Canada; email: jubernar@cs.ubc.ca; M. Hutter, TU Darmstadt, Fraunhofer Strasse 5, 64283, Darmstadt, Germany; email: igd@marco-hutter.de; M. Sedlmair, University of Stuttgart, 70569, Stuttgart, Germany; email: Michael.Sedlmair@visus.uni-stuttgart.de; M. Zeppelzauer, St. Pölten University of Applied Sciences, Matthias Corvinus-Strasse 15, 3100, St. Pölten, Austria; email: matthias.zeppelzauer@fhstp.ac.at.

2160-6455/2021/08-ART20 $15.00
https://doi.org/10.1145/3439333

Additional Key Words and Phrases: Interactive machine learning, visual-interactive labeling, active learning, candidate selection strategies, instance selection strategies, query strategies, property measures, taxonomy, machine learning explainers, visual analytics, clustering, classification

ACM Reference format:
Jürgen Bernard, Marco Hutter, Michael Sedlmair, Matthias Zeppelzauer, and Tamara Munzner. 2021. A Taxonomy of Property Measures to Unify Active Learning and Human-centered Approaches to Data Labeling. *ACM Trans. Interact. Intell. Syst.* 11, 3-4, Article 20 (August 2021), 42 pages.
https://doi.org/10.1145/3439333

1 INTRODUCTION

Labeling datasets is a precondition to conduct supervised machine learning. The fundamental idea of data labeling is to transform a data instance from unlabeled to labeled by assigning a label to it, allowing it to be used as *training data* to build models or *testing data* to evaluate built models. Manually assigning enough labels to support supervised learning in a large dataset would be expensive and time-consuming, so considerable work has been devoted to accelerating the process by reducing human involvement. The standard approach is to carefully select a small set of data instances to manually label, then propagate these labels to unlabeled instances using semi-supervised learning techniques [17, 89]. In these cases, the labeling task is split into two parts: *selecting an instance* from a set of unlabeled candidate instances, and actually *assigning label information* to that instance. In this work, we focus on the former: the problem of instance selection (also known as query selection) [118]. The scope we consider is categorical label information supporting the training of binary or multi-class classifiers across all application domains. These categorical labels are attached to instances of any type of data that can be represented with numerical feature vectors. There are many possible *instance selection strategies*: for example, concentrating on the most dense regions first or starting by selecting one instance per cluster near its centroid. In this work, we break down instance selection strategies into lower-level building blocks that we call *properties*, to support fine-grained analysis of the commonalities and differences between these strategies.

We define a property as a measurable characteristic of the data directly, or of a machine learning model applied to data. For example, the *Dense Areas First* strategy combines the properties of data density and data coverage, leading to a strategy that prefers dense data regions at different locations in the feature space [20]. The *Centroids First* strategy combines the properties of centrality and coverage to select instances located near cluster centers across the feature space. We identify a concise set of 15 primary properties that serve to cover most of the purposes followed by strategies proposed in previous work, listed in Figure 2(b). We prioritize concepts behind properties that are interpretable and well studied in the existing literature on instance selection strategies.

Instance selection strategies have been studied from two perspectives, **machine learning (ML)** and **visual analytics (VA)**. In this work, we explicitly build upon a unified concept of instance selection, combining both the ML and the VA perspective, referred to as *Instance Selection Strategies*. The ML-driven perspective is strongly shaped by **active learning (AL)** [118], which is a semi-supervised technique where the machine learner proactively asks the user for a label for a specific instance. In this case, the instance to be labeled next needs to be chosen by an algorithm, i.e., an instance selection strategy based in AL. AL is a well-known, intensively investigated, and frequently applied concept to address data labeling problems [54, 100, 118]. The properties used in AL-based strategies are straightforward to formalize, due to the wealth of references and algorithmic descriptions available in the ML literature. Recent surveys of AL strategies [54, 118] feature taxonomies with individual properties we incorporated into our set of 15 primary properties. For example, *uncertainty* is the main property of uncertainty sampling strategies, *density* of density-weighting strategies, *variance* of variance reduction strategies, and the *agreement* and *disagreement* properties are used for query-by-committee strategies.

The VA-driven perspective on instance selection strategies, which we call **human-centered (HC)**, has been explored by the visualization community. In contrast to the AL approach of automatically choosing which instance to select next, HC approaches employ visual interfaces [30, 60, 64, 115] that enable humans to directly *identify and select* the next instance to label. Visual interfaces exploit the pattern detection capabilities of human perception, allowing that knowledge to be directly expressed through wise choices of what instance to label next in a way that can immediately be exploited within ML models. HC approaches are most suitable very early in the labelling process [24], where algorithmic approaches often struggle to cold start [9]. Recent experiments showed that HC approaches can outperform AL approaches [20] in the first 50 labeling iterations, and identified 10 HC instance selection strategies some of which diverge in interesting ways from AL-driven strategies and do not align well with AL-driven taxonomies [54, 118]. While the literature on formalizing HC strategies is minimal to date [24], HC strategies appear to have complementary strengths to AL strategies [25].

We seek to understand the design space of instance selection strategies broadly, encompassing both AL and HC approaches and thereby providing a unified perspective. One motivation is to design better strategies that outperform current approaches, possibly with hybrid approaches that combine the strengths of both [25]. Another motivation is to assess the performance of instance selection strategies with respect to data and ML model characteristics; i.e., with respect to underlying properties. While the 15 properties we identify in Figure 2(b) do cover most of the design space, they do not suffice for either of these goals in terms of descriptive, generative, or evaluative power [14]. To precisely distinguish between existing strategies, design new ones, or assess how they relate to data and model characteristics, we need a systematic breakdown into a more fine-grained set of building blocks. A final motivation is to provide novel platforms for future systematic evaluation of strategy performance [80].

We propose **Property Measures (PM)** as a lower-level building block for describing the behavior of instance selection strategies: For each instance (data point) in the input dataset, a PM quantifies a specific property by assigning a numeric output value for it. This concept is very general; concrete examples include *Lowest Centroid Distance* to asses the *centrality* property of instances, *Largest Median Neighbor Distances* as a means to identify *outlierness*, or *Largest Entropy* where the probabilistic outputs of a classifier (i.e., class likelihoods) are used to highlight instances of high *uncertainty*. Virtually any existing selection strategy can be built from combinations of PMs. The explanatory power of PMs comes from considering the full space of possibilities of how they can be constructed and what purposes they can serve. We define the design space of PMs through a taxonomy that consists of four orthogonal dimensions: *Model Output*, *Instance Relations*, *Measure Functionality*, and *Measure Valence*. The main components of each dimension are summarized in Figure 2(c). In addition, the teaser figure (Figure 1) shows an overview of the connections between the four dimensions and provides details about the characteristics of individual dimensions. As we will show in the following sections, the taxonomy has enough descriptive power to distinguish PMs by their primary characteristics and enough generative power to support the design of new measures. In support of these goals, we further present a functional decomposition of the PM design space that describes PM synthesis as a cascade of eight atomic functions that allow us to represent a PM as a compact functional signature.

We created and refined the taxonomy through a highly iterative process. Our research was grounded in a thorough review of the literature for both the ML and VA perspectives on strategies for selecting instances to label, to be informed by previous taxonomies and understand their gaps

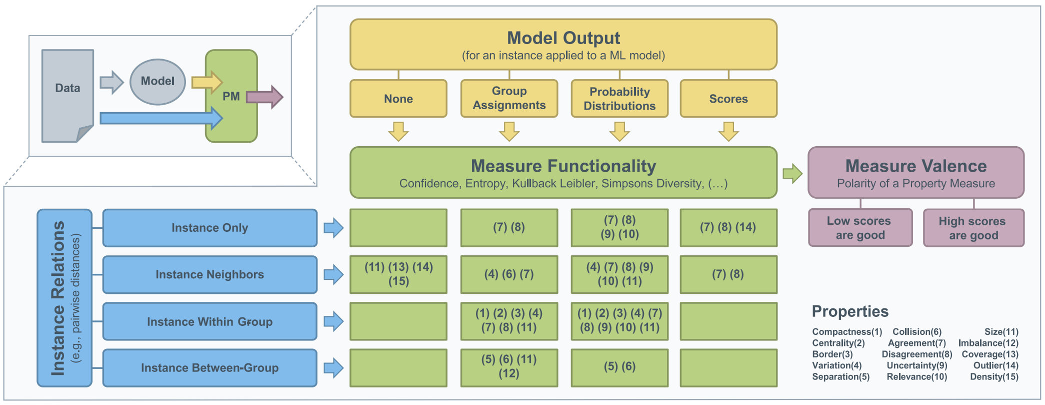


Fig. 1. Overview of our taxonomy of PM at two levels of detail. PMs measure a specific property given the *Instance Relations* of data (blue), ML *Model Output* (yellow), or both. The *Measure Functionality* (green) describes how the syntactic information that pertains to both the ML model and the data is combined. The *Measure Valence* (purple) defines the polarity of the measure. Numbers (1–15) link the legend of properties with the distribution of properties across two dimensions of the design space (blue and yellow).

in terms of descriptive, generative, and evaluative power [14]. We synthesized ideas across these rather disparate literatures to create an initial characterization of PMs, an initial taxonomy articulating their design space, and an initial task abstraction for the goals of PM designers. We were also informed by standard taxonomies of ML models, as shown in Figure 2(a) [48]. Guided by these ideas, we also created an interactive VA explainer to support the qualitative visual data analysis of PMs and the PM taxonomy through tasks that we had identified. The deeper understanding of PMs and their characteristics resulting from this analysis led us to improve the taxonomy, through many rounds of iteration: The VA explainer system helped us refine the taxonomy, and in turn as our taxonomy evolved it led us to improve the explainer and refine our task abstraction. All three research artifacts—the PM taxonomy, the explainer software, and the task abstraction—were iteratively refined and developed in parallel, with insights from one leading to the improvement of the other, over a period of two years. After many rounds of formative evaluation and refinement of intermediate versions, we used the same infrastructure for summative evaluation of the final version to provide preliminary validation of its utility. Section 6 presents this qualitative visual data analysis using the final version of the explainer with our final set of tasks to assess the final version of the taxonomy.

In summary, our research process culminated in two contributions. Our primary contribution is to establish a design space encompassing instance selection strategies from both AL and HC perspectives. We

- identify a set of 15 properties that substantially cover this space,
- define and characterize PMs that systematically quantify these properties, and
- present a taxonomy of PMs consisting of four dimensions that span the instance selection design space.

Our secondary contribution is to design and build computational infrastructure to analyze and assess the components of this design space. We

- propose eight analysis tasks to assess PMs,
- create a VA explainer to enable qualitative visual data analysis of the PMs taxonomy through the eight tasks, and

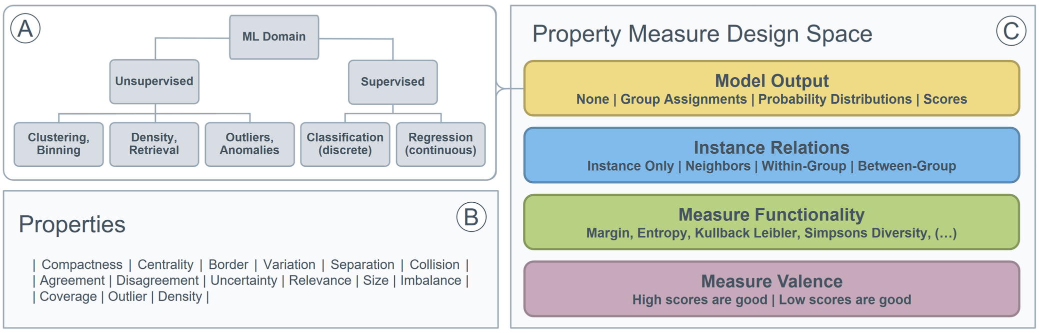

A
ML Domain
Unsupervised
Supervised
Clustering, Binning
Density, Retrieval
Outliers, Anomalies
Classification (discrete)
Regression (continuous)
Properties
B
Compactness	Centrality	Border	Variation	Separation	Collision
Agreement	Disagreement	Uncertainty	Relevance	Size	Imbalance
Coverage	Outlier	Density
Property Measure Design Space
C
Model Output
None | Group Assignments | Probability Distributions | Scores
Instance Relations
Instance Only | Neighbors | Within-Group | Between-Group
Measure Functionality
Margin, Entropy, Kullback Leibler, Simpsons Diversity, (...)
Measure Valence
High scores are good | Low scores are good

Fig. 2. (a) Summary of a ML-driven taxonomy [48]. (b) The 15 primary properties of instance selection strategies that cover ML-driven and HC-driven strategies. (c) The four dimensions of our design space for PMs are shown. Our ML *Model Output* dimension (yellow) connects PMs to the ML domain. The *Instance Relations* dimension (blue) connects PMs to the input data.

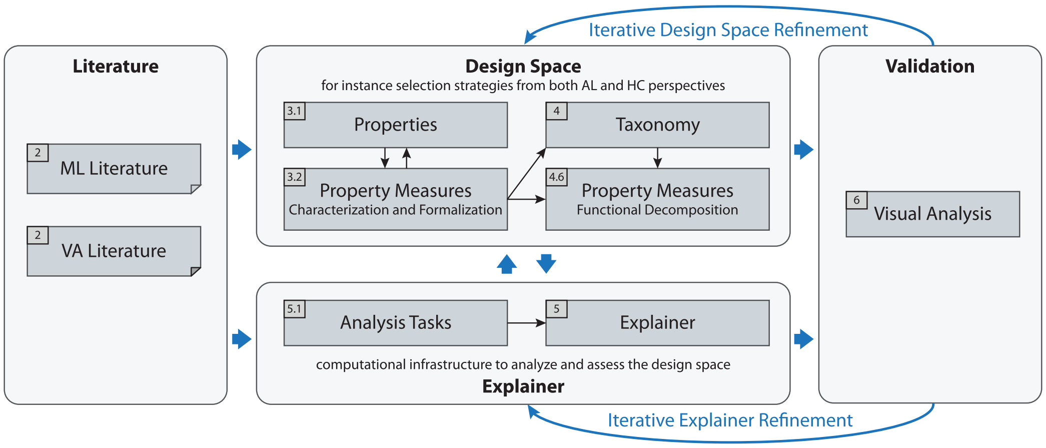

Iterative Design Space Refinement
Literature
2 ML Literature
2 VA Literature
Design Space
for instance selection strategies from both AL and HC perspectives
3.1 Properties
4 Taxonomy
3.2 Property Measures
Characterization and Formalization
4.6 Property Measures
Functional Decomposition
Validation
6 Visual Analysis
5.1 Analysis Tasks
5 Explainer
computational infrastructure to analyze and assess the design space
Explainer
Iterative Explainer Refinement

Fig. 3. Overview of the work process showing core elements and the iterative process of this research effort, with the linearization into article sections for these elements.

- implement a large class of PMs through cascades of eight atomic functions, incorporated into the explainer.

We evaluate the final versions of the PMs and the taxonomy in terms of qualitative visual data analysis via their implementations in the explainer, presenting summative evaluation using the same infrastructure that was used for formative evaluation during the iterative process of refining them. Figure 3 depicts the process and its iterative nature, showing the connections between the key elements of literature review, the development of the design space and the explainer, and the validation through visual analysis that led to the iterative refinement of these. Figure 3 also shows how we linearized these elements into sections, providing an overview of this article.

2 RELATED WORK

We discuss the related work for instance selection strategies, measures and metrics of properties, and explainable AI, considering each from both the ML and VA perspectives. The work presented here is an extended version of a workshop short paper [19] that contains preliminary ideas and first implementations; our contributions here surpass these baselines with respect to scope, extent, and the level of detail, as discussed below.

2.1 Instance Selection Strategies

Research into the selection of instances either by algorithms (AL) or by humans (HC) has evolved quite differently, as our overview of these two perspectives shows.

2.1.1 The ML Perspective. AL heavily relies on instance selection strategies for identifying the most useful instances to be labeled by a human user. AL strategies are discussed in detail in a number of surveys [54, 100, 117, 133, 137] and can be coarsely partitioned into five groups: (i) uncertainty sampling, (ii) query-by-committee, (ii) error reduction schemes, (iv) relevance-based selection, and (v) purely data-centered strategies. We briefly describe the basic intuition behind these groups. *Uncertainty sampling* aims at finding those instances that the ML model (the "learner") is most uncertain about (e.g., instances near decision boundaries) [133]; example AL strategies are *least significant confidence* [118], *smallest margin* [143], as well as *maximum entropy* [135]. In *Query by committee* [121], instances for which a committee of classifiers disagrees in prediction most are considered interesting and most helpful [92]. *Error reduction* strategies select those instances that may change the underlying classification model most. Selection criteria are expected model change [120], risk reduction [104], or variance reduction [66]. *Relevance-based* strategies [135] focus on instances with the highest probability to be relevant for a certain class. Thus, this strategy fosters the identification of positive examples. *Data-driven* strategies are independent of the supervised ML model but rather operate in an unsupervised manner. Many techniques build upon density-based or clustering-based selection criteria [117, 143], where the instances are selected from representative areas of the feature space. Density-based sampling is particularly promising for initiating an AL process when no labels are available at all, i.e., cold start problems [9, 80]. Recently, approaches toward *learning* instance selection strategies have been introduced [78]. However, this requires a series of previous AL experiments from which to draw useful conclusions. Learned strategies represent an interesting topic for future research that is not targeted explicitly in this work but that may benefit from the formalization and building blocks introduced introduced herein.

A comprehensive taxonomy of AL strategies has been introduced in Fu et al. [54]. The authors differentiate between strategies that either (i) operate on the *uncertainty of individual and independent samples* or (ii) build upon *correlations of instances*. The first group is partitioned into *Uncertainty Sampling*, *Expected Gradient Length* and *Variance Reduction*, similar to the taxonomy of Settles [117]. The second group extends existing categorizations and introduces four sub-types, namely strategies that exploit *feature correlation*, *label correlation*, *feature and label correlation*, and *structure correlation* (i.e., spatial closeness). This taxonomy mixes instance selection and label inference in the same structure and lacks detail in some branches. While being inspiring at the macro-level, the proposed categories are less useful to characterize a design space for instance selection strategies. From the literature, we observe that existing taxonomies strongly overlap and build upon *properties* of input data and ML models that are common to many different instance selection strategies. These commonalities enable a more comprehensive and better formalized description of existing strategies as presented in this article.

2.1.2 The VA Perspective. In contrast to AL, HC-based approaches enable users to select instances, by using interactive visual interfaces. Interactive labeling interfaces can, e.g., be built upon scatterplots (combined with dimensionality reduction) [16, 68, 113] or upon spatial mappings of pairwise distances of instances [23, 30]. The star coordinate interface [115] and list- or row-based interfaces [47, 75] have also been used for user-based instance selection. Finally, some approaches combine visualization techniques to enhance the instance selection informed from different perspectives [33, 84].

While powerful visualization techniques exist to enhance instance selection, the principles behind how users select instances are scarcely investigated in surveys or taxonomies. A pioneer work was presented by Seifert and Granitzer [115], who aimed at simulating and formalizing user-picking strategies. Their star coordinates interface emphasizes classifier uncertainty/certainty, enabling users to develop strategies based on *uncertainty*, *relevance*, and *variation* properties. In a user study that compared the performance of AL strategies with user-based strategies, Bernard et al. identified ten different user strategies while observing users [20]. In followup-work, the authors algorithmically formalized these 10 user-based strategies and employed them in simulated AL experiments [24]. Observed user strategies include *equal spread*, *centroids first*, *outliers first* and *class borders first*.

This article refines and extends work originally introduced in a workshop short paper [19], which proposed a preliminary taxonomy for low-level building blocks called degree-of-interest functions building upon the identified user selection strategies. That taxonomy coarsely differentiates between *data-based* and *model-based* degree-of-interest functions. Data-based functions can be based on *clustering*, *outliers*, and *density* while model-based functions describe *relevance*, *uncertainty*, spatial *relations*, *model change*, or the agreement of *committees*. The degree-of-interest functions [19] were an intermediate state in our process for what we now call *PMs*. PMs, their formalization, and the task characterization are coined by the reflection on degree-of-interest functions and by insights gained from working with these functions. A shortcoming of the preliminary taxonomy of degree-of-interest functions [19] was the tree-based structure: While it was useful for the identification of many of the 15 primary properties, the taxonomy lacked generative power for the creation of new PMs. In contrast to the previous version, the PM taxonomy we propose here defines a novel design space across four orthogonal dimensions, with enough generative and descriptive power to encompass both the ML and VA perspective. In addition, we present a functional decomposition of the PM design space, which can be used to implement all of the 605 PMs investigated in this work as cascades of eight atomic functions.

2.2 Measures and Metrics

At a high level, one difference between ML and VA is the purpose of measures and metrics. While the ML perspective is mostly used to assess properties in a non-visual way, in VA assessed properties relate to user perception and preference, using visual interfaces as a means to amplify cognition.

2.2.1 The ML Perspective. Most AL strategies build upon measures or metrics to compare or estimate data and model properties. Metrics like Manhattan distance and Euclidean distance are typical candidates for distance measurements [59, 111] between different instances or an instance and a class boundary or cluster centroid. Not all distance measures are, however metrics in a mathematical sense, e.g., cosine similarity. Another class of frequently used measures are functions for the comparison of different probability distributions, such as Kullback-Leibler divergence [85], the Kolmogorov-Smirnov test [77, 125], and the Jensen-Shannon divergence [71]. When it comes to AL strategies based on clustering [69], cluster validity, and quality measures are important, such as Dunn-like indices [49], Silhouette index [110], Davies-Bouldin measure [45], and Ward's linkage criteria [139]; see Reference [58] for an overview. Further measures that may be employed in AL strategies are graph-related measurements, such as graph centrality measures which estimate the importance of a node in a graph [27]. Related to these are measures that define distances to cluster centroids [39] to assess the centrality of instances. The above measures are important building blocks for the estimation of different properties and thus represent one dimension of the design space for PMs.

2.2.2 The VA Perspective. Measures play also an important role in data visualization. These measures are often referred to as *visual quality metrics* and their primary goal is to support an analyst in identifying "interesting" patterns in the data. The most established set of visual quality metrics are the Scagnostics measures [141], which allow to quantify patterns such as clumpiness and outlierness in scatterplots. The nine Scagnostics measures are most related to our *outlierness*, *density*, and *compactness* properties. Many others have built upon this idea, seeking to further strengthen the original Scagnostics measures [42, 93, 138], and proposing new metrics for other tasks and visual idioms [15, 41, 43, 88, 112, 124, 130]. Many of them relate to the properties that we characterize in Table 1. While Scagonstics-like measures rely on a statistical definition of patterns, another recent trend of measures has looked into modelling human perception directly. Such measures can, e.g., predict how humans perceive correlations [108] or cluster patterns [1, 10, 113] in scatterplots. These works primarily relate to *compactness* and *separation* properties. Our work relates to this line of work in that we also base our PMs on human strategies. However, our measures focus on instance selection strategies.

2.3 Explainers

Instance selection strategies and underlying properties have an inherent relation to the characteristics of the data and models involved in the labeling process. In the following, we review how explainers focused on these characteristics fit into the terrain of **explainable AI (xAI)**.

2.3.1 The ML Perspective. In ML, explanations, interpretations, and transparency are essential as AI-assisted decision systems are increasingly finding their way into the everyday lives of many people. While some models, such as decision trees are rather self-explaining, complex models such as random forests [28] and deep neural networks [87] hardly allow a direct interpretation of their behavior. The spectrum of xAI approaches is rich and heterogeneous, as can be seen in recent ML-based surveys and taxonomies [2, 8]. An important differentiation of xAI methods is into *data-* and *model-driven* explanations. Data-driven approaches focus on the analysis of data, distributions, and the extraction of interpretable features [35]. Model-specific methods aim at either explaining individual decisions (*local* methods) or entire models (*global* methods), e.g., by using *surrogate models* [147]. In contrast to focusing on either data or model explanation, our PM-based explainer needs to support both. Another differentiation is between self-explaining methods that learn to generate an explanation during model training [61] and post hoc approaches that can be applied directly to previously trained models [40, 74, 116]. With our emphasis on the interactive labeling process, requirements to our xAI tool includes aspects of both post hoc and self-explaining methods. Furthermore, our approach enables perturbation-based interpretability [31], which enables interaction in the explainability process to verify how the model responds to changes.

2.3.2 The VA Perspective. We narrow the scope to VA-based xAI methods with a particular focus on types of analyses related to our approach. We use PMs to unveil the characteristics of underlying ML models for the data labeling process to facilitate *model explanation*. Similarly to our approach, measures have been used to visually assess the separability of clustering [114], per-instance uncertainty of classifiers [107], or the characteristics of dimensionality reduction techniques [101]. Other explainers focus on the complexity of ML architectures [65] and their underlying dataflows [142], the hidden state dynamics of neural networks [127], or characteristics of classifiers such as decision boundaries [51, 90], internal tree structures [134], or performances [5, 6]. Just like our explainer, some model explanation approaches focus on the *visual comparison* of multiple models and model characteristics [4, 44, 79]. We share the idea to visually search for agreement and disagreement patterns across models. The use of xAI in the *labeling process* to identify characteristics of data and models has not been covered in previous work, although the rationale for doing so is provided by earlier observation and explanation approaches for instance selection strategies using VA techniques [18, 115]. The previous short paper [19] that we extend here is the first prototype to do xAI for the labelling process. Accompanied with the labeling process is the observation of the *learning process* of the ML model itself. Examples for the visual assessment of the model learning process include approaches for the identification of stable layers for in-depth investigation [103], the assessment of class confusions over time [62], and works studying the involvement of users to steer model building during training [128]. A final aspect refers to *what-if-analysis*, following the idea to explain characteristics by effects that have been caused by some pre-defined stimulus. Together with What-If Tool [140] and the Prospector tool [81], we share the idea to let users edit data points using a visual interface to experiment with hypothetical model conditions. While the related works focus on the manipulation of features of individual data instances, our focus is the direct manipulation of data points (in scatterplots) [19].

3 PROPERTIES AND PROPERTY MEASURES

This section describes properties and defines PMs. The introduction of properties is useful as it helps us to abstract from concrete instance selection strategies introduced in the ML and VA domain and to organize them systematically. While from the ML perspective, properties abstract certain instance selection strategies to query useful instances in the active learning process, from the VA perspective properties help to describe different user selection strategies and criteria for (visual) quality assessment. PMs represent concrete formalizations of properties and can directly be used to build instance selection strategies in ML and user selection strategies in VA.

3.1 Properties

We have identified 15 individual properties that are common to most strategies and quality evaluation measures in ML and VA. The identification of this set of 15 properties is the culmination of a long-term research effort documented by a series of publications that build on each other [18–20, 24]. These previous efforts included both computational experiments and user studies, and preliminary efforts to organize selection strategies into categories. The iterative methodology that we used to consolidate the insights from our previous work into these 15 properties is summarized in Figure 3: We combine multiple rounds of intensive literature review with multiple rounds of visual analysis using the explainer that we developed. The literature review included the analysis of existing surveys on AL strategies [54, 100, 117, 133, 137] and work on user selection strategies [20, 24, 115] to identify underlying data and model properties that are common to different strategies. The explainer incorporates implementations of many instance selection strategies from AL and many user selection strategies observed in use by humans.

A property is a measurable characteristic either of the data itself, or of a ML model output once it has been applied to data. The identified properties cover the combined space of both ML and VA approaches to instance selection strategies. Table 1 summarizes these properties, providing descriptions and examples of their use in the literature. The differentiation of ML and VA literature indicates the relevance of properties in the two domains and also reveals that some appear predominantly in only one of those fields.

We now relate these properties to influential survey works and taxonomies. With the *Compactness*, *Centrality*, *Variation*, and *Border* properties, we explicitly consider four characteristics of groupings. A grouping refers to as an assignment of instances to groups like, e.g., the results of clustering or classification techniques. To assess these four grouping-based properties, we make use of

Table 1. The 15 Properties We Identify as Central to Instance Selection Strategies, with Brief Description and the Breakdown of Related Work Pertaining to Each in Both the ML and VA Literatures

Properties	Description	ML Ref	VA Ref
Compactness	Compactness of a group of instances	[45, 49, 58, 69]	[1, 17, 113]
Centrality	Proximity of an instance to a centroid, representative, or center of gravity.	[27, 39, 58, 99, 139]	[20, 24, 34]
Border	Proximity of an instance to the outbound of a group, e.g., a cluster or class border.		[20, 24, 124]
Variation	Degree to which a group of instances varies. Assessed with measures of variance, diameter, error or loss functions for classifier internals, or diversity for distributions of class votes.	[54, 66, 71, 77, 85, 98, 122, 123, 125, 139]	[1, 10, 73, 113]
Separation	Separability of a group of instances from other groups. Can be at the granularity of instances or groups.	[45, 49, 58, 69, 110, 114, 139]	[10, 17, 106, 114]
Collision	Degree to which a group collides with other groups. Measures use the overlap of (fuzzy) clusters, confusion of class distributions, or crosscut of modeled topics.	[12, 26, 126]	[114, 124]
Agreement	Agreement of predictions or assignments made by a model for a set of (related/nearby) instances.	[55, 71, 77, 85, 125]	
Disagreement	Disagreement of predictions or assignments made by a model for a set of (related/nearby) instances	[55, 71, 77, 85, 92, 121, 125]	
Uncertainty	Exploitation of uncertainty information, e.g., provided with probabilistic classifiers or assessed by distances to decision boundaries.	[118, 118, 119, 132, 143]	[115]
Relevance	Exploitation of certainty or confidence information to select instances that are likely to be relevant. Also used to confirm or reinforce class information structures.	[11, 57, 135–137]	[17, 20, 115]
Size	Amount/count of instances within a group.	[146]	[114]
Imbalance	Comparison of an observed distribution of labeled instances across classes with an expected distribution of labeled instances. Used to identify and mitigate imbalance of class label distributions.	[13, 70, 100, 145, 146]	
Coverage	Distribution of a set of instances across the entire space. Used to select a diverse set of instances across the high-dimensional feature space.	[54, 73]	[20, 22, 24]
Outlierness	Focus on instances that with abnormal or even unique characteristics. Assessed by outlier/anomaly detection models or inverted density estimations.	[29, 32, 76, 97, 105, 141]	[20, 24, 113]
Density	Identification of dense regions of a dataset.	[94, 129, 141]	[20, 24]

a rich body of measures and metrics that has been presented to support ML tasks [58, 66, 69]. However, AL taxonomies consider clustering rather coarsely as one type of *data-driven* [54, 118] strategies. In contrast, HC strategies can be mapped to these four properties more explicitly. According to the observational study by Bernard et al. and the replication study by Chegini et al. [20, 34], users apply grouping-based properties explicitly when labeling cluster patterns (*centroids first*, *cluster borders first*) or class patterns (*class borders refinement* and *class distribution minimization*). Finally, Benato et al. [17] observed a HC strategy where users focus on compact groups first for label propagation in semi-supervised ML.

The two opposed properties *Separation* and *Collision* support the characterization of between-group relations (clustering results, classification results, etc.). While in ML a variety of measures for these properties are provided [12, 26, 58, 126], AL strategies based on group separation or collision are rare. Research in VA underlines the *Separation* and *Collision* properties in two ways: First, the emphasis on visual quality measures for cluster and class separation measures [10, 114] shows the relevance for the HC perspective. Second, given the observed *class intersection minimization* [20] HC strategy and the user tendency to focus on mixed groups [17], users explicitly focus on separation and collision characteristics, which indicates the need for *local class separation* and *local class diversity* assessment to support the labeling process [24].

The two opposed properties *Agreement* and *Disagreement* directly build upon *query-by-committee* AL strategies [54, 100, 118]. In general, we use these properties to characterize the divergence [71, 77, 85, 125] of a set of model outputs (ensemble) or similar sets of outputs.

With *Uncertainty* and the *Relevance* property, we account for two frequently applied strategies in AL: *uncertainty sampling* and *relevance-based sampling* [54, 100, 118]. Seifert and Granitzer focused on HC strategies [115] based on uncertainty, supported with a visual interface that emphasizes uncertain class assignments.

Size is a universal property that can be applied to any set of instances. As such, it nicely complements other data characteristics in both AL and HC contexts.

The *Imbalance* property is inspired by AL strategies that aim at balancing the distribution of labels among classes [13, 70, 100, 145, 146]. However, HC strategies have not yet been formalized.

Coverage addresses unexplored regions in the feature or data space, a characteristic that is relevant for AL and HC. In AL, coverage strategies [22] are considered beneficial in early stages of the labeling process [54]. Similarly, HC strategies also employ the so-called *spatial balancing* [24] principle (users focus on various data regions early).

Outlierness is a property that is opposed to most other data-centered properties. While having a strong background in ML [3], AL strategies often aim at avoiding outliers, especially in early phases of the labeling process [118]. In HC, users seem to apply at least two strategies related to outlierness: *outliers first* and *class outlier labeling* [20].

Density concludes our list of primary properties. In AL, density-based instance selection strategies (density weighting, density-based sampling) are defined in most taxonomies [54, 100, 118]. Similarly, HC strategies include the frequently applied *dense areas first* [20] strategy, which seems to be very effective in early labeling iterations [24].

3.2 Property Measures

3.2.1 Definition of Property Measures. A PM quantifies a property in a systematic way. For every instance of a given dataset the PM assigns a number to it.

PMs capture properties of instance selection strategies; conversely, PMs can be combined to form strategies. These basic properties can be calculated from multivariate datasets directly (e.g., density), as well as from the output of ML models (e.g., class separation). PMs can be applied to the output of a broad spectrum of ML models. Figure 2 provides an overview of principal classes of ML techniques and describes the interaction between PMs and models in the ML domain. PMs receive instances as input and calculate a numerical score for any instance as output. A typical dataflow of PMs is shown in Figure 4: Both the data instances and the output of an ML model serve as input to a PM, so characteristics of both may be reflected in the PM output. The output scores for any instance can be used for downstream applications. For the selection of instances to be labeled next, ranking of PM output scores helps to prioritize instances according to the corresponding property. Normalization of PM output scores, as indicated by the purple arrow in Figure 4, may be necessary when combining PMs together within instance selection strategies. A concrete example

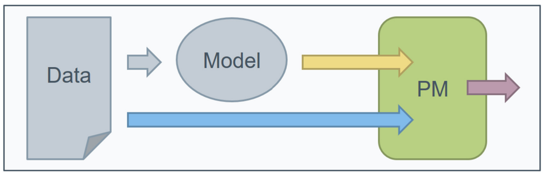


Fig. 4. Interaction of PMs with datasets and ML models. PMs measure a specific property given the input of data (blue), ML model output (yellow), or both. The output of a PM (purple) can be used for downstream analyses.

of PMs are centrality-based PMs, which assign individual centrality values to any instance using distance metrics measuring the *centrality* of instances within a grouping such as the output of a cluster or classifier.

3.2.2 Property Measures as Functions. In the following, we provide a more formal definition of PMs, considered as functions. A dataset X is a set of instances x that are high-dimensional feature vectors; that is, each instance is represented using an n-dimensional real vector $x \in \mathbb{R}^n$. Our goal is to define a set of functions that assign a real value v to any possible instance x, even to instances that are not contained in the original dataset. These functions are supposed to describe arbitrary characteristics (that is, *measures*) for instances, usually in relation to other instances of the dataset. A model M is (optionally) applied on X, and its output can be used by the measure function to calculate the output for an instance. In the most generic form, such a measure function f can be described as

$$f : \mathbb{R}^n \to \mathbb{R}$$
$$f(X, x, M(X, x)) \mapsto v.$$

4 A TAXONOMY OF PROPERTY MEASURES

We present a taxonomy of PMs that defines the design space in terms of four orthogonal dimensions: *Model Output*, *Instance Relations*, *Measure Functionality*, and *Measure Valence*. We can order these with respect to the flow of data through the PM into three stages, as shown in Figure 5: PM input, PM internal, and PM output. The input stage consists of the *Model Output* dimension (yellow), which incorporates everything that is provided upstream of a PM and is used as-is by it. The PM-internal stage encompasses the relations between data instances considered as feature vectors in a high-dimensional space (blue), and the *Measure Functionality* dimension (green) that integrates calculations about the *Model Output* and data *Instance Relations*. The output stage incorporates the *Measure Valence* dimension that concludes the processing of individual PM scores for downstream usage (purple). A typical downstream application would be to use the output PM score for the selection of instances in an AL system. We now we describe these four dimensions in detail, in this dataflow order.

4.1 Model Output

The design decision in the input stage of the dataflow through a PM is to determine what type of ML *Model Output* is used. The standard hierarchical taxonomy of models used in the ML domain, shown in Figure 2 (upper left) [48], is a useful way to reason about the intended purpose and usage context of a model, starting the fundamental split into unsupervised and supervised ML. In the unsupervised case the models are used for clustering and binning to group instances together, density and retrieval to identify similar instances, or outlier and anomaly detection to

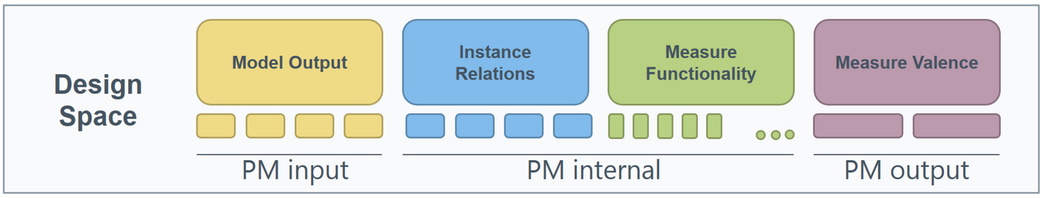


Fig. 5. The four orthogonal dimensions of the PM taxonomy, ordered according to three stages of dataflow. The input stage pertains to the *Model Output* dimension that contains four sub-categories, the internal stage has both the *Instance Relations* dimension with four sub-categories and the *Measure Functionality* dimension with many sub-categories, and the output stage covers the *Measure Valence* with two sub-categories.

find unusual instances. In the supervised case the models are used for classification with discrete data, or regression with continuous case.

At the highest level, a shared characteristic of all of these models is that they produce output, which in turn is the input to a PM. We analyzed the kinds of output produced by these models in terms of syntactic characteristics, leading us to propose four *Model Output* sub-categories that cross-cut the standard hierarchy:

- **None:** The simplest case handles PMs that do not exploit any ML model output. These PMs can be calculated solely with data-centered measure functionality. Representative properties are assessments of *density* and *coverage*, which can be calculated by solely using pairwise distances.
- **Group Assignments:** Instances are grouped into crisp subsets through approaches such as classification, clustering, binning, or user selection. The syntax of the ML model output is a categorical attribute indicating which group is assigned to an instance. These assignments are also referred to as votes in classification. Properties at play in this category include *size*, *compactness*, *variation*, *separation*, *confusion*, *agreement*, or *disagreement*.
- **Probability Distributions:** Instances are grouped into soft assignments for example from probabilistic classifiers or fuzzy clusterings. The syntax of the ML model output for an instance is not just a single item, but rather an entire array of numerical values (probabilities), one for each group. Probabilities are required by PMs that support properties such as *uncertainty* or *relevance*.
- **Numbers/Scores:** A quantitative value is expressed as a single number for every instance, as for example provided with outlier analysis algorithms or regression models.

Our formalism encompasses singleton models as well as *ensemble* models, in a unified way. For example, the PM implementations we use in the explainer include seven classifiers and an eighth ensemble classifier that incorporates all seven of them.

4.2 Instance Relations

The *Instance Relations* dimension is the first PM-internal stage (cf. Figure 5). The dimension describes the structural relationships between one instance and other instances in the high-dimensional feature vector space, as used in the PM. We subdivide this dimension into four categories, as shown in Figure 6, according to what information the PM exploits:

- **Instance Only:** only the information that is provided with the focus instance
- **Instance Neighbors:** information stemming from the focus instance and from instances in its local spatial region
- **Instance Within-Group:** the information of instances within a pre-defined group
- **Instance Between-Group:** the information across multiple pre-defined groups of instances

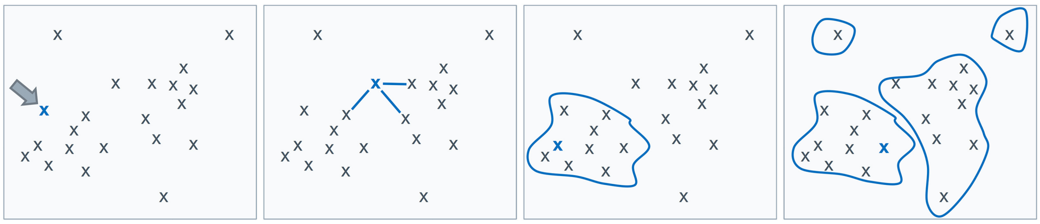


Fig. 6. Instance Relations of PMs (blue dimension). This structural dimension describes which types of data relations PMs use. Every PM corresponds to one of four categories: instance only, instance neighbors, instance within-group, and instance between-group (left to right).

This dimension requires that information is provided to the PM about multiple instances in addition to the specific focus instance, except for the degenerate instance-only category. The last two categories also require information about pre-defined groups, which arises within ML models only from certain types. These models are exactly the ones found within the *Group Assignments* and *Probability Distributions* categories of the *Model Output* dimension, so there is an interplay that crosscuts between these two dimensions. Groups could also be created in an HC setting through interactive user selections. The information about all of the relevant interests is passed on to the next dataflow stage, the *Measure Functionality* dimension, to perform computations such as pairwise similarity measures or distance metrics between the instance feature vectors.

4.3 Measure Functionality

The *Measure Functionality* dimension is the second PM-internal dimension, describing how information that pertains to both the *Model Output* and *Instance Relations* dimensions is combined according to the syntactic structure defined by the sub-categories of each.

In contrast to the low-cardinality categorization of the other dimensions, this dimension has a large number of categories. These fall into two high-level sets, unary aggregator functions that compute a single number, and pairwise measure functions where the computation encompasses two instances. Frequently used functions are listed below:

- **Unary (Aggregation) Functions:** $\mathbb{R}^n \rightarrow \mathbb{R}$
  - *Statistical Aggregation:* uses statistics (such as min, max, median) to extract single numbers from a set of numbers
  - *Probability Aggregation:* uses measures (such as entropy [122], output margin [143], confidence [118], GiniIndex [56]) to extract single numbers from a set of probabilities
  - *Distance Aggregation:* uses measures and metrics (such as Dunn-like index [49], Silhouette index [110], diameter) to extract a number from a set of distances
  - *Integer Aggregation:* uses measures (such as cardinality, difference, count) to extract an integer number from a set of integers
  - *Diversity Aggregation:* uses measures (such as Simpsons diversity index [123]) to extract a number from a set of integers
- **Binary (Measure) Functions:** $\mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$
  - *Distance Measure:* using distance metrics (such as the Euclidean distance metric or the cosine similarity) to receive a distance between two instances
  - *Divergence Measure:* using divergence measures (such as Jensen-Shannon [71], Kullback-Leibler [85], or Kolmogorov-Smirnov [77, 125]) to receive a divergence/disagreement between two instances

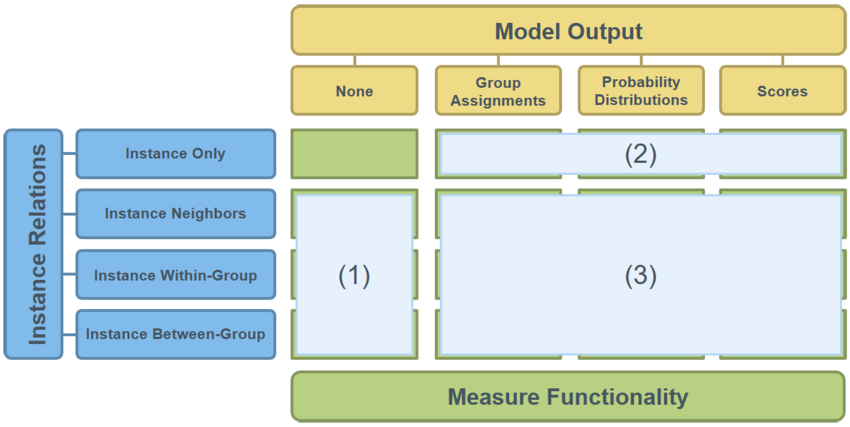


Fig. 7. Three types of examples showing how PMs use the information that pertains to *Model Output* and *Instance Relations*.

Figure 7 shows three types of examples to demonstrate how unary and binary functions are used by PMs. The first type (1) calculates instance relations (e.g., pairwise distances between instances). The relations can then be used to apply unary aggregation functions such as distance aggregations or statistical aggregation. Properties that can directly be addressed in this way are, e.g., *density*, *coverage*, our *outlierness*.

The second type of examples (2) only aggregates the model output to receive an output value for any instance. Prominent properties are (a) *uncertainty*, supported by PMs using probability aggregations such as output margin [143], entropy [135], or confidence [118] measures to condense probability distributions or (b) *outlierness* supported by PMs using scores provided with outlier analysis algorithms [3].

The third type of examples (3) incorporates both distance information from instance relations as well as a model output. This is the most complex type of input information that has to be conveyed and by the measure functionality. One solution to cope with this complexity is using distance information from adjacent instances as a *weighting* criterion for the distillation of the model output. A straight-forward approach would be to take the output of close neighbors stronger into account than the output of more distant instances. Examples include weighting the (e.g., Kullback Leibler) divergence measure results of related instances, or weighting the (Simpsons) diversity of class assignments (votes) of instance neighbors.

Even if framed by the semantics of the property and the syntax of the given input, PM implementations may still rely on various internal functionalities. Especially for the third type of approach, the *Measure Functionality* of a PM may combine different functions, metrics, or heuristics to aggregate/condense the input information down to a single numerical value.

4.4 Measure Valence

The dimension of *Measure Valence* determines the polarity of a PM, i.e., if the ordered values of a PM output are interpreted from low to high or from high to low, i.e., which is considered better, larger or smaller numbers? In practice, the valence of measures is often determined by the downstream task at hand. To have full control over the polarity, we explicitly define the *Measure Valence* as the fourth dimension of the design space.

There are numerous properties that can be considered from diametrical perspectives. For example, the *agreement* property is the opposite of *disagreement*. Another example is the prominent AL strategy smallest output margin [118] based on the *uncertainty* property. In contrast, selecting the instance with the largest output margin picks the most certain instance, shifting the focus to the *relevance* property.

Having this dimension explicitly within the taxonomy allows a simpler structure for the *Measure Functionality* dimension, so that no duplication is required. The *Measure Valence* dimension can also be used in a way similar to normalization, for example to flip a PM to produce a range from 0 to 1 instead of from 1 to 0. With the *Measure Valence* dimension, we increase the generative power of our taxonomy, as it doubles the PM design space.

4.5 Mapping Properties across Taxonomy Dimensions

We consider the crosscut between the two central dimensions of the taxonomy, *Model Output* and *Instance Relations*, to show how properties distribute across those two dimensions of the design space.

First, we assess the distribution of properties into the 4×4 grid formed by the subcategories of each of those two discrete dimensions. Figure 8 shows the distribution of all 15 properties (Table 1) using numbers (1–15) to link between the legend at the lower left and occurrences in the 4×4 grid (center, green). The analysis of the PM distribution reveals several insights. First, the descriptive power of the taxonomy lies in the fact that different grid cells have noticeably different set of properties associated with them, so our proposal does not simply mirror the structure suggested in previous work. Second, not every cell of the grid is populated with properties. These gaps show the generative power of our taxonomy: They suggest possible locations for novel yet unexplored properties. Third, we observe that the properties distribute well into individual cells, rows or columns in the design space, e.g., the pure data-driven properties 14 and 15 appear only on the first column while property 1 is typically in the third row (within group comparison). This kind of distribution of properties across the design space shows that the dimensions have been chosen in a reasonable way. The outlierness property (15) is the only one located at two non-adjacent cells. We explain this fact by noting that outlierness can be implemented in two ways: using an outlier ML model (top right) or by exploiting instance relations (left). Fourth, classical AL properties based on uncertainty (9) and relevance (10) only populate a small fraction of the design space (3 of 16 cells). Alternative properties based on other data and model characteristics exist, which may compliment the strengths of classical AL properties. Fifth, the instance between-group properties separation (5), collision (6), and imbalance (12) clearly separate from remaining properties. One reason for this separation is the different working practice of between-group PMs compared to other PMs. Finally, even if no model output is provided, four different instance neighbors-based properties (size, coverage, outlierness, density) can be implemented. This purely data-centered approach complements the rationales of many model-centric AL strategies.

Next, we assess the property distribution using the tree visualization in the explainer (T7). We construct a hierarchy using the *Model Output* dimension first, followed by the *Instance Relations* dimension, and then the PM properties for a third layer. The resulting tree, shown in Figure 9, shows a structured overview of the mapping from PMs to properties. One visually salient feature is that PMs assigned within same sub-trees are highly similar, whereas PMs in different sub-trees vary considerably in most cases. Another is that properties are not equally distributed across the space defined by the two primary levels of the hierarchy (the *Model Output* and *Instance Relations* dimensions of the taxonomy). We see that multiple properties fall within with the same combination of categories of the two dimensions. In particular, the combination *Group Assignments* and *Instance Within-Group* contains six of our 15 primary properties.

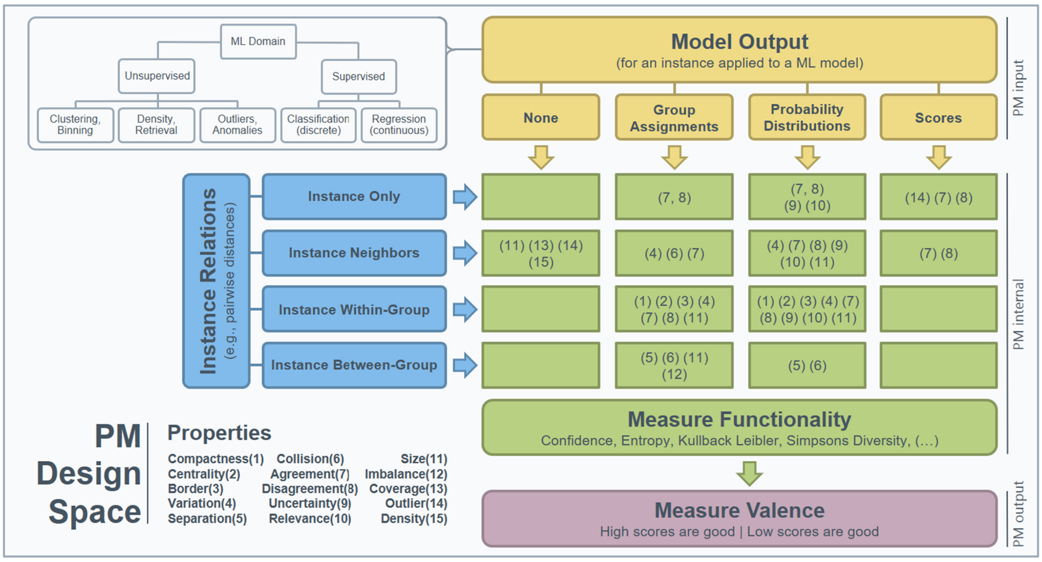


Fig. 8. Design space for PMs using the four orthogonal dimensions to characterize PMs. Following the dataflow, the *Model Output* dimension (yellow) defines the input of the PM. As such, the *Model Output* is the anchor to the ML domain. Crosscutting *Model Output* with *Instance Relations* (blue) is particularly well suited to discriminate properties structurally, i.e., in a grid-based arrangement. *Measure Functionality* (green) and *Measure Valence* (purple) form the remaining two dimensions of the design space, completing the dataflow.

4.6 Functional Decomposition of Property Measures

4.6.1 Decomposition into Atomic Functions. Guided by the dimensions of the taxonomy, we decomposed PMs into a set of eight atomic functions that simplify implementation. These eight functions suffice to implement all of the PMs used in the explainer.

The functions are described here conceptually, primarily differentiated according to what data types they operate on instances, sets of instances, numbers, or sets of numbers. Therefore, many of these functions can be applied to the results of other functions, so they can be composed into a cascade. Table 2 provides an overview of the eight functions, including a brief description.

4.6.2 PM Creation Using the Decomposition. Given the decomposition into atomic functions, different PMs can be assembled by composing these functions. We propose the use of compact *signatures* to compactly describe the computation of PMs [96]. Signatures enable to quickly grasp the computational composition of a PM and enable their structural comparison. We show the composition of PMs addressing the properties (1) *uncertainty* (with a least significant confidence PM), (2) *density* (using lowest median neighbor distances), (3) *compactness* (identifying the lowest cluster diameter), (4) *disagreement* (of a classifier ensemble assessed with the Jensen-Shannon divergence), and (5) *coverage* (using highest minimum neighbor distances to already labeled instances). We briefly describe how the decomposition to 8 atomic functions can be used to create these PMs. Figure 10 provides details for the five composition examples, aligns the five implementations with the characteristics of the design space, and shows the five PMs in the explainer. It can be seen that the characteristics of the implemented PMs distribute differently across the design space.

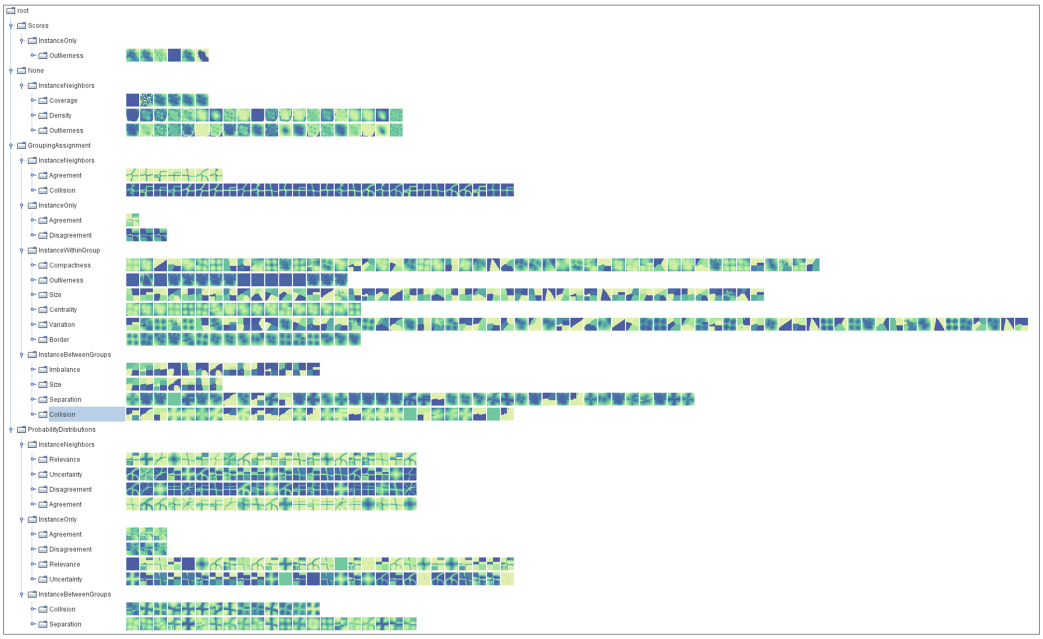


Fig. 9. Tree visualization of the explainer used to distribute the 15 properties of 605 implemented PMs across the *Model Output* and the *Instance Relations* dimension. Most implementations use group assignments or probability distributions of upstream ML models. The highest cardinality of properties in one crosscut of the two dimensions is six for *GroupAssignment* versus *InstanceWithinGroup*.

Uncertainty: Least Significant Confidence: Input is a classifier that reports class probabilities for a given instance. The PM estimates the uncertainty of these predictions by computing the maximum (i.e., if there is no outstanding probability (confidence), the class prediction comes with a high uncertainty). Using the atomic functions, the classifier represents a soft *Assignment* L, returning class probabilities. An *Aggregation* A is used to compute the (maximum). Finally, with the max-min *Normalization* N, instances with least confidences will yield highest PM output values, leading to the functional signature $PM(x) : L \rightarrow A \rightarrow N$.

Density: *Lowest Median of Neighbor Distances.* To calculate a density property, we calculate the lowest median distances of instance neighbors. A first step is the retrieval of a subset in the vicinity of an instance using a *Selection* R. A *Pairwise Measure* D is applied to estimate the distance between the current instance and all its neighbors. A statistical *Aggregation* function A returns the median of the pairwise distances. The max-min *Normalization* N provides density results by inverting the value domain of the PM outputs. The functional signature of this distance-based PM is $PM(x) : S \rightarrow D^* \rightarrow A \rightarrow N$. The * indicates that function D is applied several times in the computation, i.e., for each instance in the selection S.

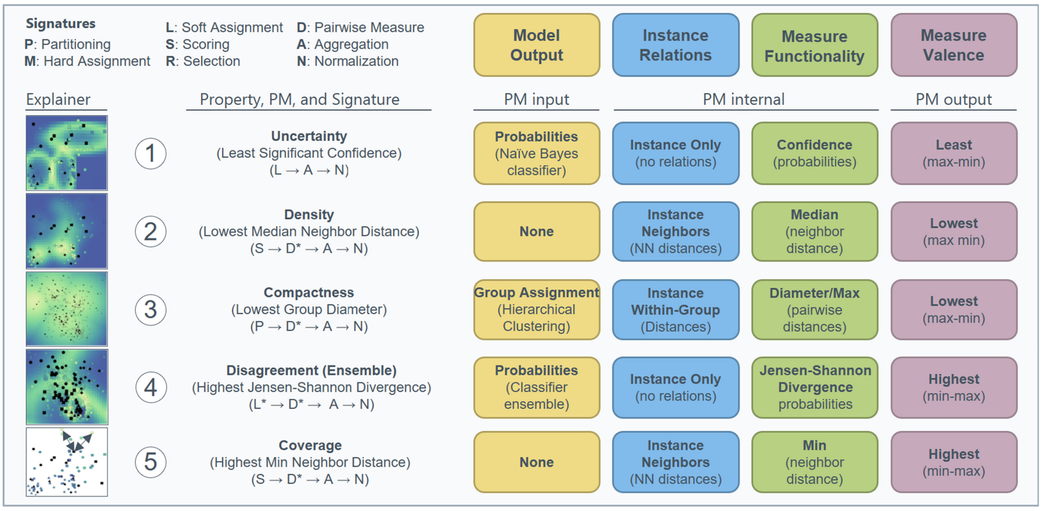
Compactness: *Lowest Group Diameter.* One way to asses the compactness of a group of instances is applying statistics on pairwise distances. Given a *Partitioning* P and a *Pairwise Measure* D (distance metric), the distances can be calculated. A statistical *Aggregation* function A (maximum) applied on the pairwise distances provides the diameter of the group The max-min *Normalization* N leads to compactness results by inverting the value domain. The functional signature of this compactness-based PM is as follows: $PM(x) : P \rightarrow D^* \rightarrow N$

Table 2. Decomposition of PMs into a Set of Eight Atomic Functions That Simplify PM Implementations

Function		**Description**
Partitioning	P	Maps instances of a given set to non-empty, disjoint subsets of that given set. Examples are clusterers, classifiers, binning functions, or alternatively interactions of users such as lasso selections.
Hard Assignment	M	Receives an instance, and returns a categorical identifier representing a certain group the given instance belongs to. In our context, this identifier may indicate a grouping that was created with a partitioning function.
Soft Assignment	L	Receives an instance, and returns a probability distribution representing the likelihood of the instance to belong to one of k groups (e.g., classes or clusters). Soft assignment functions use algorithms such as probabilistic classifiers or fuzzy clusterers
Scoring	S	Receives an instance, and computes a single numerical result. The scoring function may know (or be trained on) a data context X. Two classes of ML models that implement such a function are regression models and outlier analysis algorithms.
Selection	R	Takes an instance and a set of instances, and returns a subset of the given set. Examples include retrieval operations revealing nearest neighbors or instances within an epsilon range.
Pairwise Measure	D	Bi-function applied to two sets of numbers that computes a single numerical result. This function can be a distance metric, but can also be correlation-, diversity-, or divergence measure.
Aggregation	A	Applied to a set of elements, returns an element of the same type. This may be applied to a set of instances, e.g., to receive the centroid of a group of instances. An aggregation can also be used to condense a set of real numbers using statistics (e.g., min, max, mean, median).
Normalization	N	Performs a re-scaling of numbers. When the range for the normalization is known, a normalization can re-scale a number to the 0-1 range. More general forms of normalization functions compute the minimum and maximum range internally, requiring sequence of numbers as input.

Disagreement: *Highest Divergence of a Probabilistic Classifier Ensemble.* The disagreement of classifier ensembles is an important criterion of the selection of instances. The PM uses the soft **Assignment** L of multiple probabilistic classifiers (L^*) and applies a *Pairwise Measure D* (Jensen-Shannon Divergence) and a statistical *Aggregation* function A (mean) to compute the disagreement. A min-max *Normalization N* completes the functional signature of this disagreement-based PM: $PM(x) : L^* \rightarrow D^* \rightarrow A \rightarrow N$

Coverage: *Highest Minimum Neighbor Distance.* Especially in early iterations of the labeling process, it is relevant to achieve an equal spread of the training data. The PM does not require a model output but retrieves a subset in the vicinity of an instance using a *Selection R*. The PM is special in that only instances of the training data are retrieved. By this means, it is possible to allocate new regions in the space that have not been labeled yet. A *Pairwise Measure D* (distance metric) reveals distances to training data, a statistical *Aggregation* function A returns the minimum distance of the instance to the training data. An obligatory min-max *Normalization N* completes the functional signature of this coverage-based PM $PM(x) : S \rightarrow D^* \rightarrow A \rightarrow N$.



Signatures
P: Partitioning **L**: Soft Assignment **D**: Pairwise Measure
M: Hard Assignment **S**: Scoring **A**: Aggregation
R: Selection **N**: Normalization

Explainer		Property, PM, and Signature	Model Output (PM input)	Instance Relations (PM internal)	Measure Functionality (PM internal)	Measure Valence (PM output)
	1	**Uncertainty** (Least Significant Confidence) (L → A → N)	**Probabilities** (Naïve Bayes classifier)	**Instance Only** (no relations)	**Confidence** (probabilities)	**Least** (max-min)
	2	**Density** (Lowest Median Neighbor Distance) (S → D* → A → N)	**None**	**Instance Neighbors** (NN distances)	**Median** (neighbor distance)	**Lowest** (max min)
	3	**Compactness** (Lowest Group Diameter) (P → D* → A → N)	**Group Assignment** (Hierarchical Clustering)	**Instance Within-Group** (Distances)	**Diameter/Max** (pairwise distances)	**Lowest** (max-min)
	4	**Disagreement (Ensemble)** (Highest Jensen-Shannon Divergence) (L* → D* → A → N)	**Probabilities** (Classifier ensemble)	**Instance Only** (no relations)	**Jensen-Shannon Divergence** probabilities	**Highest** (min-max)
	5	**Coverage** (Highest Min Neighbor Distance) (S → D* → A → N)	**None**	**Instance Neighbors** (NN distances)	**Min** (neighbor distance)	**Highest** (min-max)

Fig. 10. Characterization and functional decomposition walkthrough for five sample PMs using the taxonomy (cf. Section 4) and the atomic functions (cf. Section 4.6). For replicability, this information is provided for all 605 PMs in a supplemental materials table. (1) An uncertainty-based PM uses the probabilistic output of a Naïve Bayes classifier. The measure functionality calculates the confidence (highest probability), and the max-min normalization valence assigns highest values to lowest confidences. Bright colors in the explainer for least significant confidences indicate classifier decision boundaries. (2) A density-based PM uses instance neighbors (nearest neighbor distances) and calculates the median neighbor distances. Valence (max-min) inverts the value domain of PM outputs, so bright regions in the explainer show lowest values. (3) A compactness-based PM uses the model output of a hierarchical clustering. Using the instance-within group functionality, the measure functionality of the PM calculates the maximum of all pairwise distances (diameter). Valence (max-min) inverts the PM outputs, revealing most compact groupings. (4) The disagreement of a classifier is calculated using the probabilistic outputs of a classifier ensemble, applying the Jensen-Shannon divergence measure to assess the disagreement across the ensemble committee. Bright explainer colors show uncertainty, namely ensemble disagreement. (5) A PM based on data coverage identifies instances in the unlabeled data that are far away from training set instances. No model output is required, nearest neighbor distances to the training data are calculated, and the minimum distance is taken. The instance with the highest minimum neighbor distance is most relevant, having the largest distance to the training set.

4.7 Implementations

For our analyses, we implemented 605 PMs in total, Figure 1 provides an overview. To ease the implementation of PMs, we introduced and defined PMs formally in Section 3.2 and drew a connection of PMs to their functional characteristic with pre-defined inputs and a numerical output. In addition, in Section 4.6.1, we presented the functional decomposition of PMs into eight atomic functions that allow the implementation of all PMs used in the explainer. In Section 4.6.2, we demonstrated the applicability of the functional decomposition using five PMs with different characteristics as examples.

In the following, we briefly outline the classes of ML models and model algorithms we used for our PM implementations, before we provide details about the 605 concrete PM implementations. Details about the functional decomposition and the combinatorics provided with the design space for PM implementations based on individual ML models are provided in the supplemental materials.

4.7.1 ML Model Implementations. The *Model Output* dimension of the taxonomy opens the design space for principal classes of ML models used in the ML domain. The syntax of the *Model*

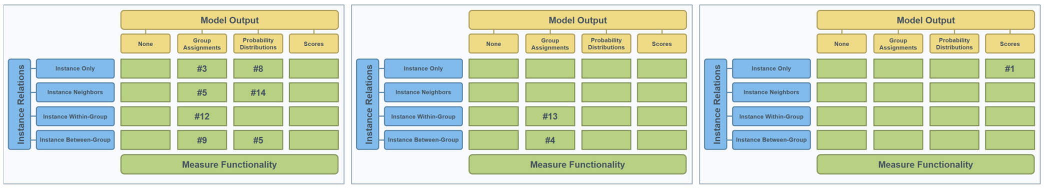


Fig. 11. Distribution of PM implementations across the *Model Output* and *Instance Relations* dimensions (yellow and blue). Left: Fifty-six PM implementations can be created with a single classifier. At a glance, seven types of measure functionalities (green) are used to compose these 56 implementations. Center: Seventeen PMs are composed by a single clustering algorithm. Right: Currently an outlier algorithm forms a single PM.

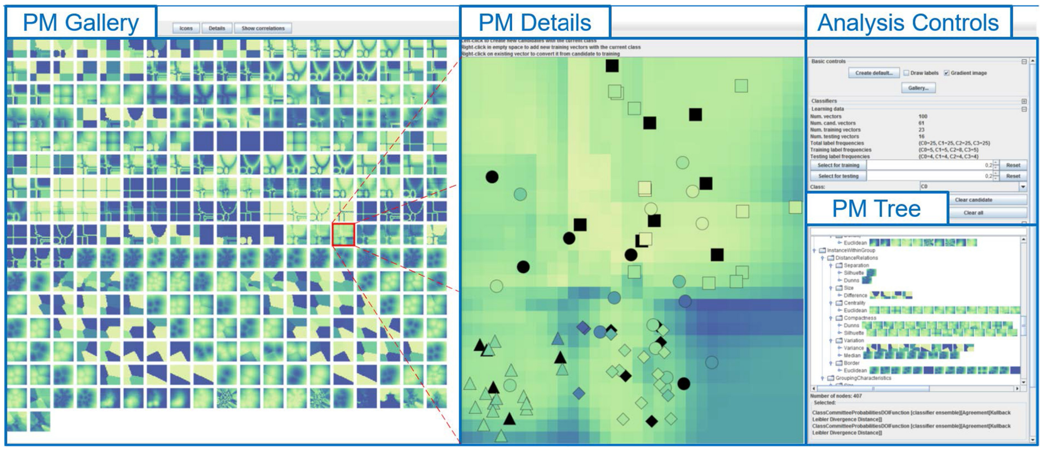
Table 3. Algorithms for Classification, Clustering, and Outlier Detection Used for the Implementation of PMs

Classifier Algorithms		**Clustering Algorithms**		**Outlier Algorithms**	
NaiveBayes	[48]	AffinityPropagation	[52]	AngleBasedOutlierDetection	[82]
KStar	[37]	Canopy	[95]	DistanceBasedOutlierAnalysis	[76]
BayesNet	[53]	ExpectationMaximization	[46]	DynamicWindowOutlierFactor	[97]
RandomForest	[28]	FarthestFirst	[63]	KNNOutlierAnalysis	[105]
MultilayerPerceptron	[67]	SimpleKMeans	[7]	LocalIsolationCoefficient	[144]
SVM	[38]	XMeans	[102]	LocalOutlierFactor	[29]
SimpleLogistic	[86]	HierarchicalClustering (CompleteLinkage)	[139]		
		HierarchicalClustering (SingleLinkage)	[139]		
		HierarchicalClustering (AverageLinkage)	[139]		

Output categories (*Group Assignments*, *Probability Distributions*, and *Scores*), determines which ML models can be used. In the following, we briefly list the concrete ML algorithms we used to implement and instantiate the 605 PMs in Table 3. The selection of involved algorithms does not claim to be exhaustive. Rather it was our goal to use multiple models for either category. Criteria have been the heterogeneity of ML models to see different effects by using PMs, as well as their popularity and relevance in the ML domain to provide an intuitive access to the models and their internals.

We used the three classes of ML models to compose 605 PMs. Details about PM implementations and about how we used the combinatorics of the dimensions of the design space are descried in the supplemental materials. Figure 11 gives a brief overview of the distribution of PM implementations across the *Model Output* (yellow) and *Instance Relations* dimensions (blue) for PMs based on a single classifier (left), a single clustering algorithm (center), and a single outlier algorithm (right). In the supplemental materials, we describe the distribution of PMs across the design space in detail.

4.7.2 PM Implementations. Figure 10 walks through only 5 PMs. In service of replicability, we provide full details about all 605 implemented PMs with a table in the supplemental materials that exactly characterizes every PM implementation according to the four dimensions of the design space and the associated property. In the table, there is one row for each 605 implementation and the four dimensions of the design space form the columns. In addition, the property of every PM implementation forms a fifth column. As an example, the *smallest margin* [143] AL strategy is created with a PM that uses (1) *probability distributions* as *Model Output*, (2) *instance only* as



Left-click to create new candidates with the current class
Right-click in empty space to add new training vectors with the current class
Right-click on existing vector to convert it from candidate to training

Num. vectors 100
Num. cand. vectors 61
Num. training vectors 23
Num. testing vectors 16
Total label frequencies (C0=25, C1=25, C2=25, C3=25)
Training label frequencies (C0=5, C1=5, C2=8, C3=5)
Testing label frequencies (C0=4, C1=4, C2=4, C3=4)
Select for training 0,2
Select for testing 0,2
Class: C0

Number of nodes: 407
Selected:
ClassCommitteeProbabilitiesDOIFunction [classifier ensemble][Agreement[Kullback Leibler Divergence Distance]]
ClassCommitteeProbabilitiesDOIFunction [classifier ensemble][Agreement[Kullback Leibler Divergence Distance]]

Fig. 12. Prototype for the interactive *explainer* of data characteristics and ML models using PMs. The PM Gallery View (left) provides an overview of the outputs of all implemented PMs. Currently, one PM is selected (red rectangle). Details of the selected PM can be analyzed in the PM Details View (center), selection interaction indicated with red dashed lines. Users can activate the data space coloring mode or the instance coloring mode, or both using the analysis controls panel (upper right). Other controls allow steering training and testing data, as well as classifier selection and training. The PM Tree (lower right) enables structuring PMs according to hierarchies or taxonomies and can be used as an alternative way to select PMs. In this example, a Demo Dataset was used (cf. Section 6.1).

Instance Relations, (3) *margin* as *Measure Functionality*, and (4) *smallest* as *Measure Valence*. Its entry in the supplemental table is ProbabilityDistributions[MultilayerPerceptron classifier], InstanceOnly[none], Smallest, Margin. This table provides developers with an exact recipe to replicate the 605 PM implementations in their programming language of choice.

5 AN EXPLAINER FOR PROPERTY MEASURES USING VISUAL ANALYTICS

We built a visual explainer tool, referred to as the *explainer* below, to support the analysis of PMs. We first describe the eight analysis tasks for PMs in Section 5.1, and then describe the three main views of the explainer in the remainder of this section. We then demonstrate the use of the explainer to conduct visual qualitative data analysis through these tasks in Section 6. Our choice to take a visual analytics approach was motivated by the iterative and incremental nature of HC and AL, where the interplay between humans and ML models takes place within a feedback loop. Moreover, the explainer was crucial in refining the PM taxonomy. The explainer is the result of an iterative design, refinement, and extension process. The preliminary version, presented in the workshop short paper that we extend [19], included only a prototype for the *PM Detail View*.

Figure 12 shows an overview of the explainer. The main views are the *PM Details View* (center), the *PM Gallery View* (left), the *PM Tree View* (lower right), as well as an *Analysis Controls View* (upper right). In the following, we present the design of each main view in terms of both visual encoding and interaction designs.

5.1 Tasks for Analyzing Property Measures

We develop the formalisms of properties and PMs as a means, toward the end of articulating a design space of instance selection strategies that encompasses both ML and VA approaches in a unified way. The final taxonomy that we propose in Section 4 is the result of extensive analysis built around these PMs. We now present a set of eight tasks that comprise that PM analysis process.

Table 4. Systematic Characterization of Tasks, Each of Which Would Benefit from Visual Explanation Support

Task	Description	PM #	PM groups #	Model #	Data #	Time #
T1	Analyze data characteristics in detail	1	1	1	1	1
T2	Compare across models	1	1	n	1	1
T3	Manipulate data for what-if analysis	1	1	1	n	1
T4	Change over time while labeling	1	1	1	i	i
T5	Compare few PMs in detail	m	1	1	1	1
T6	Compare few PMs over time	m	1	1	1	i
T7	Overview across all PMs	all	1	1	1	1
T8	Compare subgroups of PMs	all	n	1	1	1

These eight tasks were identified through the same iterative methodology that we used to develop the properties, PMs, and the taxonomy itself: the combination of intensive literature review and the use of the explainer for visual analysis. Just as the explainer and the taxonomy were each used to refine the other, the analysis tasks themselves were refined in conjunction with the development of the explainer that supported them.

PMs provide a means to analyze instance selection strategies systematically and with precision, taking into account the characteristics of the input dataset and of the output of the ML model tied to a PM. The labelling process also has a temporal nature when considered as an incremental series of steps, where each assignment of a new label to an instance constitutes a timestep.

We describe the analysis tasks based on five principal aspects:

- **Data#:** the number of datasets involved
- **Model#:** the number of ML models
- **Time#:** the number of timesteps considered
- **PM#:** the number of PMs used to fulfil the task
- **PM group#:** the number of PM sets used to fulfil the task

Table 4 presents the eight tasks in terms of these five aspects. Every task (T1–T8) corresponds to a different type of information need we observed in the iterative process of analyzing PMs. Every information need addressed a unique combination of singles/multiples of (1) PMs, (2) PM groups, (3) ML models, (4) datasets, and (5) timesteps. T1 focuses on the detailed analysis of data characteristics in the most constrained scope: a single PM, ML model, dataset, and timestep. It can unveil properties about the data such as *Density* or *Outlierness.* T2 pertains to comparing *n* different ML models to each other in detail, in the context of the same PM and data, again for a single timestep. T3 involves directly changing the data instances, for example to assess the sensitivity of the ML model to these changes; we consider each of *n* changes to be a mutation of the data, in contrast to the other tasks where the data itself stays fixed. Variants of changing data are adding unlabeled instances, or modifying instances. T4 supports the analysis of the iterative labeling process over time, where one unlabeled instance is changed to a labeled instance at each of *i* timesteps (iterations). We also consider this task to involve changes of the data. T5 extends the scope from inspecting the results from one PM to comparing between *m* PMs, with all other aspects held fixed. T6 pertains to comparing both across PMs and across timesteps, analogously to T4. T7 is the overview task with the scope of analysis across all available PMs, to understand their similarities and differences broadly. Finally, T8 supports comparison between designated groups of PMs, for example subgroups that correspond to a proposed taxonomy, to check whether it is aligned with visually apparent characteristics of PMs. This task may take place when seeking evidence that a

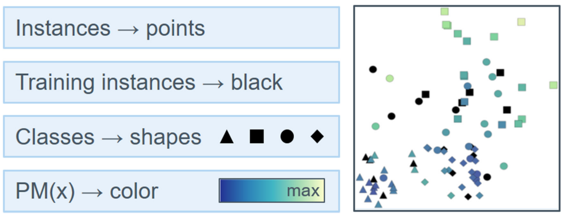


Fig. 13. Visual encodings used in the explainer.

taxonomy under consideration is valid, or to discover local regions of the PM design space suitable for preservation in a new version of a taxonomy.

5.2 PM Details View

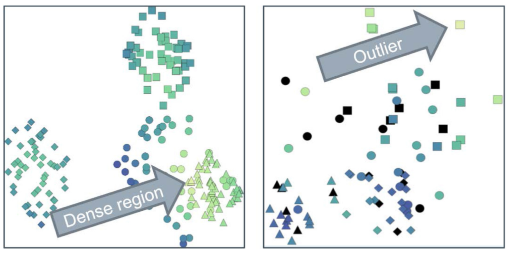
The central view of the explainer is the *PM Detail View*, showing characteristics of data and model at a high level of detail for a single PM. We use scatterplots to represent instances in two dimensions (2D). Multidimensional data is mapped to the scatterplot using dimensionality reduction techniques, where the selection of the dimensionality reduction technique is a user parameter. Default techniques in our approach are PCA [72] as a linear as well as non-metric MDS [83] and t-SNE [91] as non-linear dimensionality reduction variants.

5.2.1 Visual Representation of Data. Every visual mark in the scatterplot corresponds to one instance of the dataset (see Figure 13). We use shape to encode class information and color to encode PM values. The dataset used for illustration purposes in this article contains four classes, shown with four types of shapes (rectangle, square, triangle, and diamond). Although in other systems it is common to use color for class labels, in our case we use the color channel to show PM values. Labeled instances are a special case, because there are no PM values for them, since they already have labels assigned, and are colored black. Accordingly, a black diamond shape represents one instance of class "diamond" of the labeled training set. In contrast, unlabeled instances are the data subset that is of particular interest in instance selection and labeling tasks. For this purpose, PMs are always applied to unlabeled instances x to assess their relevance to be labeled next. We use the unipolar colormap by Tominski et al. [131] to map the continuous numerical scores of PM outputs to colors. By default, dark colors represent low PM output values; the colormap can be inverted with a control parameter. The figures in this article use the default design, so bright instances are most relevant (high PM output).

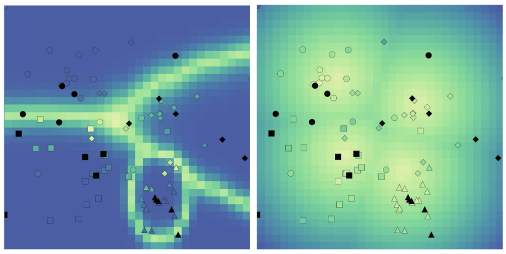
The explainer has a control for operating in *ground-truth mode*, where ground truth label information (the shapes) is shown even for unlabeled instances. This capability is useful for observational studies and ground-truth experiments, in particular for the analysis of instance selection strategies over time as in tasks T4 and T6. The figures in this article use ground-truth mode, showing the true label information as shapes also for unlabeled (colored) instances.

5.2.2 Coloring Modes. The *Details View* has two modes of color coding, *Instance Space Coloring* and *Data Space Coloring*.

The *Instance Coloring* mode is well suited for tasks T1, focusing on data characteristics in detail for a single PM and model, and tasks T4 and T6, showing changes over time as data instances are labelled. We directly color the data points with respect to their PM values [18, 19]. This approach is possible for 2D datasets (e.g., for PM validation) and for multivariate data in combination with dimensionality reduction. The examples in Figure 14(a) show how the Instance Coloring mode can be used to assess important data characteristics such as dense regions or outliers. It provides a very



(a) Analysis of data characteristics with the Instance Coloring mode, supporting T1. Left: a density-based PM supports the quick assessment of density relations in a dataset. Right: an outlier-based PM emphasizes outlier instances of a dataset.



(b) Explanation of two model characteristics (T2) with the Data Space Coloring mode. Left: a least significant confidence PM unveils decision boundaries of a Naïve Bayes classifier. Right: a centrality measure PM reveals structures of a kMeans clustering.

Fig. 14. The Details View has two modes of color coding, Instance Space Coloring and Data Space Coloring. In the four examples Demo Datasets have been used (cf. Section 6.1).

detailed understanding for the behavior of individual instances, but requires substantial screen space for the view. We demonstrate the analysis using the instance coloring mode in Section 6.2, using Figure 17 as an example.

In *Data Space Coloring* mode, the explainer computes the PM across the entire visual (output) space; that is, every pixel is considered a virtual instance and for which the PM is applied. The result is a (quasi) continuous spectrum of PM values over the entire space, which we can easily map to background colors, similar to techniques for the visualization of classifier decision boundaries [90]. If the identical normalization is used, then Data Space Coloring and Instance Coloring yield identical colors. Data Space Coloring is particularly well suited for tasks involving many PMs (T7 and T8), because it works well at small scales, and also supports many of other tasks including comparing ML models (T2), what-if analysis of changing instance locations (T3), and comparing PMs (T5).

Data Space Coloring is currently supported for all 2D datasets (instances mapped to pixels), and for multivariate data with linear dimensionality reduction (PCA). It would be straightforward to support any nonlinear dimensionality reduction techniques using methods that provide an inverse mapping for any projection [51, 109]. The Data Space Coloring mode is used in Figure 14(b) to explain model structures such as decision boundaries of a classifier (Naïve Bayes) or cluster regions (kMeans), supporting T2.

5.2.3 *Interaction Design.*

The *Detail* view enables different types of analyses (T1–T4), some of which further benefit from additional user-interaction support.

For T2 (compare across models), we support switching the ML model that is used for a single PM. A particularly relevant example for AL-based instance selection is the assessment of differences and commonalities between classifiers. The *Analysis Controls View* provides a control for the algorithm selection to switch classifiers, as an example for the selection of concrete algorithms from a set of algorithms of a ML model class. We demonstrate the comparative visual analysis of different models in Section 6.3, using Figure 18 as an example.

For T3 (manipulate data for what-if-analysis), we provide two alternative interaction designs. First, the explainer allows users to trigger hypothetical data manipulations and then observe the model changes that would result from such changes. To do so, users can edit data directly by interactively dragging instances, which triggers an instant recalculation of all models and PMs allowing users to immediately assess the effects caused by these data changes. Just like the Data

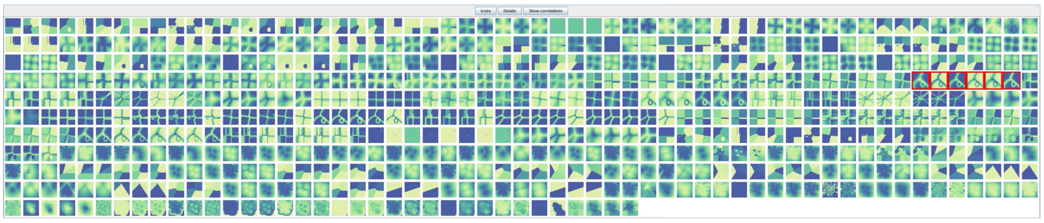


Fig. 15. PM Gallery View providing an overview of PMs. The gallery reveals the various properties that data and models can have. Six PMs have been selected showing an interesting characteristic. The detailed analysis shows that an underlying Naïve Bayes classifier produced this output. The six PMs reveal either the uncertainty or the relevance property for instance selection and thus show inverse appearance.

Space Coloring mode, this interaction is suitable for both 2D and multivariate datasets. The drag-and-drop interaction works for multivariate data in combination with dimensionality reduction techniques, however only those that exhibit an inverse mapping so that the user's changes in 2D can be mapped back to the multi-dimensional space [51]. We demonstrate the analysis using data manipulation in Section 6.4, using Figure 19 as an example. The second interaction design supports adding new unlabeled data to facilitate yet another type of what-if-analysis. A right-click automatically adds an unlabeled data point, and the ground truth label information of the new instance can be pre-defined in the *Analysis Controls View*. Similarly to data manipulation, this interaction supports the identification and explanation of model responses to data changes.

For T4 (change over time), users can interactively select instances from the unlabeled candidate set and assign them to the labeled training set. In the ground-truth mode where the labels (shapes) of unlabeled data are shown, a single click on an instance *selects and labels* a focus instance automatically. The increase of the training set automatically triggers re-training involved ML models (see Section 4.7 for an overview of model implementations). In addition, the explainer shows the output of refreshed PMs based on the updated model results automatically. We demonstrate the analysis of change over time in Section 6.5, using Figure 20 as an example. Although the explainer was primarily created in support of the research goal of studying the characteristics of instance selection strategies, this capability for interactive feedback also has utility for carrying out labeling in practice.

5.3 PM Gallery View

The *PM Gallery View* on the left in Figure 12 enables users to analyze all available PM implementations at a glance (605 PMs in our case). Small icons, one for every PM, provide a full overview of all PMs and thus ease the assessment of similarities among PMs (T7). The interface design applies a small-multiples concept to show all PMs side-by-side in a grid. Every PM is represented as a rectangular icon of equal size showing a small version of the interface used for the *PM Details View* (cf. Section 5.2). The Data Space Coloring mode is the default, because it is well suited to reveal model characteristics (cf. Figure 14(b)) and scales well to small screen spaces. The order of PMs within the grid is adjustable. PMs can be sorted based on the name of PMs, or pre-defined structural characteristics such as dimensions of taxonomies (cf. Section 4). Figure 15 demonstrates the abilities of the gallery to provide an overview of all PMs and as a starting point for the analysis of PMs in detail. In our analyses in Section 6.8, we demonstrate how we used the gallery shown in Figure 23 to gain an overview of all our PM implementations.

5.3.1 Interaction Design. There are several ways to interact with the *PM Gallery View*: Mouse-over functionality shows a tool-tip to highlight metadata about PMs such as name, description,

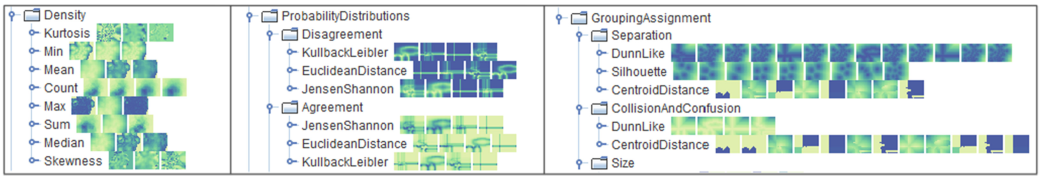


Fig. 16. PM Tree View used to show three groupings of PM subsets according to their properties at the top level and more detailed PM functionality at lower levels. Left: Density-based PMs differ by their statistical aggregation functionality. Middle: Probability-based PMs based on different divergence measures. Right: Grouping assignment PMs differ by the compactness and separation measures used.

or performance results (if the explainer is coupled with an evaluation process). The *details* button enlarges every PM icon and displays available metadata on top of each PM as a second semi-transparent layer. Users can select PMs of interest, either single PMs (the global selection model will trigger a refresh in the *PM Details View*) or multiple PMs, for example to foster the visual comparison of PM subsets (T5), which may also be analyzed throughout a labeling process (T6). To support comparing a few PMs in detail (T5), the selected subset can be displayed in a *PM Gallery View* showing PM icons at larger size with the details on. Example analyses based on T5 and T6 are demonstrated in the Sections 6.6 and 6.7 using the Figures 21 and 22. Selected PMs are always highlighted with a red bold outline. Example uses of the gallery include Figure 12 where a single PM was selected and Figure 23 in Section 6.8 (T7), where six similar PMs have been selected and analyzed in detail.

5.4 PM Tree View

With the *PM Tree View* shown at the lower right in Figure 12, we have designed a visualization that is capable of representing hierarchical partitions of PMs where individual PMs are shown at the leaf level. The tree visualization supports gaining an overview of PMs according to some designated hierarchical structure (T7), and supports the visual comparison of subgroups of PMs. An example of an overview task using the tree view is shown in Section 6.8. An obvious use case was the sub-division of PMs according to selected dimensions of our taxonomy (T8), as a means for insight generation, taxonomy validation, and refinement. In particular, the tree view helped to validate dimensions of our taxonomy in terms of its descriptive power; that is, their ability to accurately demarcate a meaningful set of existing PMs. We note that the dimensions of our taxonomy provided in Section 4 are orthogonal, so the taxonomy does not have an inherent tree structure. However, the selection and prioritization of individual dimensions within a hierarchy supports taxonomy construction and comparison, by investigating whether different top-level partitions impose visible structure in the tree view. The flexibility to select and re-order dimensions provides a dynamic and context-specific way to compare between designated subgroups of PMs, and their alignment with the taxonomy (T8). In summary, the tree is able to depict relations between PMs and external categorizations, such as the dimensions of our taxonomy.

5.4.1 Visualization and Interaction Design. To ease the readability of textual PM descriptions the *PM Tree View* is laid out horizontally. PMs of the same category are aligned side-by-side as a row of icons, similar to a line of PMs in the *PM Gallery View*. The icons in each row are also vertically aligned to facilitate comparison between lines. Sub-trees can be interactively expanded and collapsed to show the desired level of local substructure. Selections of single PMs, multiple PMs, or all PMs in sub-trees are supported, each triggering the global PM selection action just as in the details and gallery views. Figure 16 shows how sub-structures revealed by tree interaction can be analyzed in detail. The examples also give an indication for the assessment of PM subgroups with the taxonomy alignment, which is demonstrated in Section 6.9 using Figure 26.

6 QUALITATIVE VISUAL DATA ANALYSIS

We provide insight into the various types of analyses we conducted throughout the iterative process of elaborating properties and PM characteristics, and designing the taxonomy. In the following, we demonstrate examples showing how we used the explainer to perform analysis tasks T1–T8 (cf. Section 5.1). In addition to the in-depth analysis and comparison of PMs, the tasks can support the better understanding of the data and the trained ML models, as well as the behavior of HC-based and AL-based instance selection strategies. While large parts of our analyses were conducted in a formative manner, our focus in this section is to put PM implementations into action, to validate properties, PM characteristics, and the taxonomy summatively.

6.1 Datasets

We now describe the datasets used in our visual analysis process and in this section. We created synthetic data that was targeted to support different types of analyses.

For all examples of summative analyses shown in this article, we fixed two data synthesis parameters, whereas other synthesis parameters remained flexible. We fixed the number of classes to four. With four classes, we explicitly address data labeling tasks that go beyond simple binary classification (like cats and dogs). Also, in the PM Details View these four classes will still be visually discernible with the shape encoding (cf. Section 5.2). The second fixed data synthesis parameter regards the dimensionality of the datasets, which is always two. This enables us to code the data exactly in the way as they are shown in the explainer. Thus, we explicitly de-emphasize the analysis of effects stemming from dimensionality reduction mapping errors. The two dataset synthesis parameters that vary across different analysis are the distribution of instances of the same class versus the distribution of classes of different classes, following the idea to have control over the separation of classes [21].

With this setup, we characterize two types of datasets that have been used to demonstrate every PM analysis task:

- **Default Datasets:** with 400 instances, equally balanced instances per class (100 per class), various degrees of class intersections between any pair of classes.
- **Demo Datasets:** with 100 instances, equally balanced instances per class (25 per class), various degrees of class intersections between any pair of classes. Mainly used when intensive user interaction and/or data manipulation was applied (T2, T3, and T4).

6.2 T1—Analyze Data Characteristics in Detail

Figure 17 shows explainer output illustrating how PMs can unveil important data characteristics. Figure 17(a) shows the loaded dataset before any PM is applied. What can be assessed is the distribution of four classes (shapes), which are partially intersecting. We use a first PM to assess the *density* characteristics of the dataset. The PM is based on instance neighbor distances and calculates the average (statistical aggregation). In Figure 17(b), it can be seen that four dense regions in the dataset stand out. These regions match the class distribution of the data quite well. Figure 17(c) uses an *outlierness* PM, where the outlier-based ML model [76] output provides outlier scores that are directly be used by the PM where every instance is either assessed as an outlier or not. We then turn the analysis toward cluster characteristics of the dataset and thus bring another unsupervised ML model class into play. The PM in Figure 17(d) shows the *centrality* of instances according to the centroid distances of a XMeans clustering. The XMeans clustering has created two groups, as

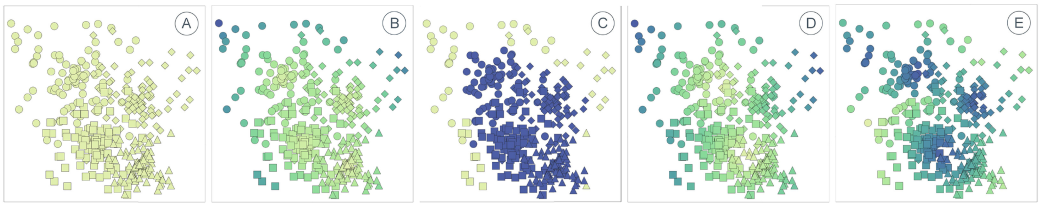


Fig. 17. Instance Coloring mode for the detailed analysis of data characteristics (T1) using a Default Dataset. (a) Blank dataset visualized in the explainer. The distribution of data and four classes (shapes) can be analyzed. (b) A *density* PM is selected, revealing four dense regions. These regions match well with the distribution of the four classes. (c) A PM based on *outlierness* uses a outlier analysis model with a binary output, assigning instances either to outliers or not. (d) A *centrality* PM highlights instances near to the center of two clusters calculated by a XMeans clustering model. (e) A *border* PM emphasizes instances at cluster borders of a hierarchical clustering.

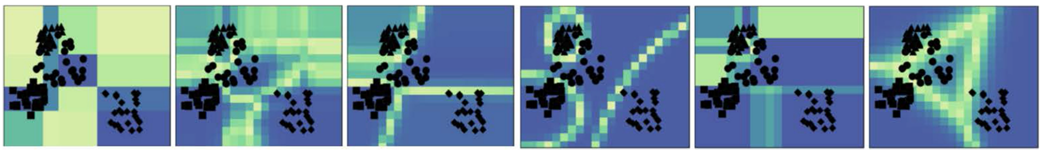


Fig. 18. Data Space Coloring mode used for the visual comparison of the probabilistic output of classifiers with an Entropy-based uncertainty PM (T2). In the scenario we assess the relation of the PM to six different classifiers (from left to right: BayesNet, KStar, Multilayer Perceptron, Naïve Bayes, Random Forest, Simple Logistic). The PM reveals regions in the data space where classifiers are unsure; in this case, the decision boundaries. The visualization helps to understand the investigated property of the PM and unveils interesting intrinsic properties of the classifiers. A Demo Dataset is used.

the PM values point toward two major regions in the dataset. The number of two groups, however, is opposed to the four classes of the dataset. While this discrepancy raises questions as to whether the clustering was able to reflect the semantics depicted with the class labels, it seems that XMeans did not define the two clusters arbitrarily: Each cluster contains two classes almost exactly. Still, the example gives an indication that ML often comes with challenges related to the choice of model parameters. In fact, the explainer enabled us to identify these characteristics about data (and models) in detail. As a final step, we take the *borders* of a clustering into account. The PM in Figure 17(e) uses a hierarchical average link clustering and calculates the centroid distances to assess border characteristics. The clustering result formed four clusters, and highlighted border instances, respectively. It can be observed that the size of the four clusters differs considerably: In contrast to a huge cluster in the lower part are three rather small clusters at the top. One may infer that the hierarchical clustering is sensitive to outliers, which would explain the very small cluster at the upper left. The analysis of data characteristics also showed the influence on ML models on the assessment of such characteristics. In particular, we observed that diverse clustering and classification algorithms can compute considerably different model outputs.

6.3 T2—Compare across Different Models

We now use the explainer for a detailed visual comparison of different ML models using the same PM and dataset. This usage scenario is motivated by the assumption that the output of classifiers may differ considerably, leading to different results of PMs.

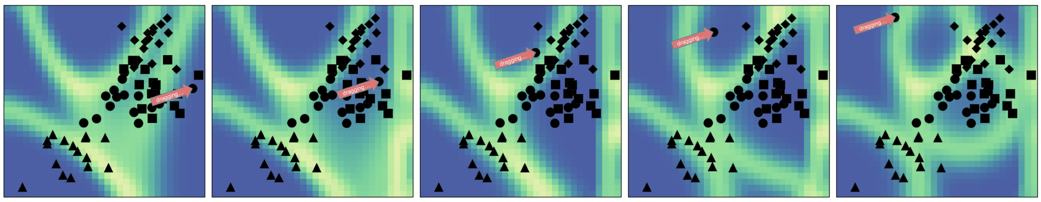


Fig. 19. What-if-analysis to better understand the interplay of PM and model (T3) for an uncertainty-based PM using Entropy, applied with a Naïve Bayes classifier: moving a single data instance of a Demo Dataset (drag-and-drop highlighted with a red arrow) causes a considerable change of the decision boundaries of the classifier. Similar effects can be observed for other combinations of PMs and ML models as well.

Using the control panel in the explainer (cf. Figure 12), users can select between different classifiers. In our scenario, we iterate over six classifiers (BayesNet, KStar, Multilayer Perceptron, Naïve Bayes, Random Forest, and Simple Logistic), the resulting *PM Detail View* is shown in Figure 18. With the Data Space Coloring mode, the visual comparison helps to unveil considerable differences between classifiers. Given that the same *uncertainty* PM was used for the explanation of the classifiers' outputs, we affirm that the choice of classification model has a considerable effect on the PM output, a characteristics that also plays an important role for many existing instance selection strategies [118].

6.4 T3—Manipulate Data for What-If-Analysis

Figure 19 illustrates how Data Space Coloring in combination with a PM (Entropy in this case) can be used to visualize and explain classifier internals such as the decision boundaries. Data manipulation is used to show how the classifier reacts to changes in the data. In the example a single sample is moved from the right border to the upper left, causing a considerable change of the decision boundaries. Note that this approach is classifier-agnostic and can be used to visualize classification boundaries of arbitrary classifiers.

6.5 T4—Change over Time While Labeling

Figure 20 demonstrates the iterations 1–4 of an *interactive labeling process* for a single PM. It can be seen that every change in the training data has a considerable effect on the output of the PM. In this example, the PM highlights the least significant confidence of a probabilistic classifier, indicating that the predictions of the classifier change substantially during the first iterations. The Instance Coloring Mode is chosen to highlight instances that would be selected next (brightest colors). In iteration 1 the instances highlighted by the PM are rather arbitrary. The reason is simple: the PM relies on the output of the probabilistic classifier that was trained with only a single instance (black, diamond class). With the selection of a second instance after iteration 2, the situation is different: The classifier partitions the dataset into diamonds and triangles, and the uncertainty-based PM reveals a clear axis of uncertainty in between (highlighted with a dashed line), which indicates the decision boundary of the classifier. The selection of a third instance (circle) again changes the classifier prediction considerably: Now three class structures are visible. After having selected a fourth instance (square), all four classes have been labeled exactly once. Again, the result of the classifier has changed considerably.

6.6 T5—Compare Few PMs in Detail

The explainer provides different ways to identify interesting subsets of PMs (cf. Section 5). We now compare a subset of a few PMs in detail. This analysis was motivated by the fact that for a single

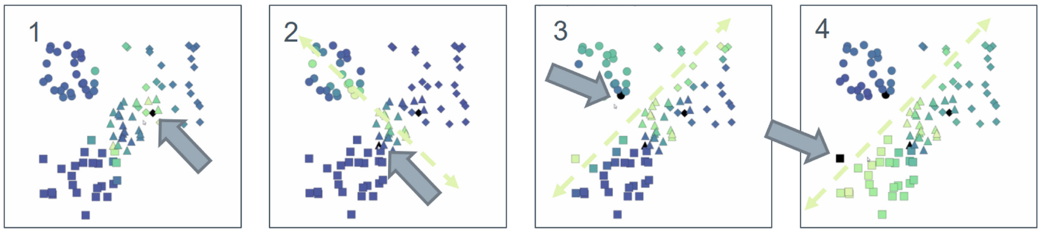
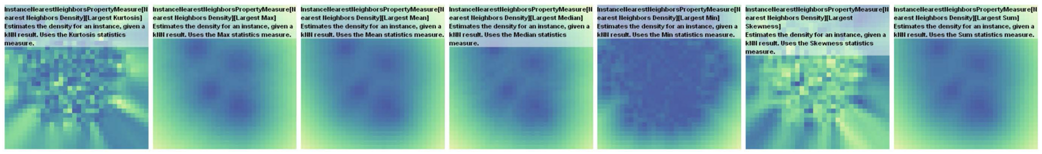


Fig. 20. Analysis of a data labeling process (T4, iterations 1–4), gray arrows mark the instances labeled per iteration for a Demo Dataset; labeled instances were added to the training set, successively. A least significant confidence PM is observed, emphasizing instances close to the decision boundaries of a classifier. The decision boundaries are highlighted by dashed, bright lines. The observation of the labeling process reveals the significant change of the model across the iterations, explained by changing PM scores.



InstanceNearestNeighborsPropertyMeasure[Nearest Neighbors Density][Largest Kurtosis] Estimates the density for an instance, given a kNN result. Uses the Kurtosis statistics measure.

InstanceNearestNeighborsPropertyMeasure[Nearest Neighbors Density][Largest Max] Estimates the density for an instance, given a kNN result. Uses the Max statistics measure.

InstanceNearestNeighborsPropertyMeasure[Nearest Neighbors Density][Largest Mean] Estimates the density for an instance, given a kNN result. Uses the Mean statistics measure.

InstanceNearestNeighborsPropertyMeasure[Nearest Neighbors Density][Largest Median] Estimates the density for an instance, given a kNN result. Uses the Median statistics measure.

InstanceNearestNeighborsPropertyMeasure[Nearest Neighbors Density][Largest Min] Estimates the density for an instance, given a kNN result. Uses the Min statistics measure.

InstanceNearestNeighborsPropertyMeasure[Nearest Neighbors Density][Largest Skewness] Estimates the density for an instance, given a kNN result. Uses the Skewness statistics measure.

InstanceNearestNeighborsPropertyMeasure[Nearest Neighbors Density][Largest Sum] Estimates the density for an instance, given a kNN result. Uses the Sum statistics measure.

Fig. 21. Visual comparison of some PMs in detail (T5). All PMs aim for the same property, use the same *Instance Relations* implementation, but differ in their measure functionality (statistical aggregation): from left to right kurtosis, max, mean, median, min, skewness, sum are used by the PMs. Kurtosis and skewness differ considerably from the remaining PMs. At the top of every PM visualization, context information can be displayed on demand, like PM descriptions or the characterization by the dimensions of the PM taxonomy.

property (density) for the same instance neighbors strategy (kNN) and the same measure valence "high is good" (min-max normalization), two of the seven PMs appear to be very different, as it can be seen in Figure 21. The explanation was provided with the PM Gallery, showing enlarged visualizations of the PM output as well as textual descriptions of the PMs. The PMs differ in their measurement functionality, i.e., in the way statistical aggregation is applied to condense the nearest neighbor distances to a single number. From left to right, the PMs use (1) kurtosis, (2) max, (3) mean, (4) median, (5) min, (6) skewness, and (7) sum. The detailed analysis reveals that (1) kurtosis and (6) skewness look considerably different compared to the remaining five statistical aggregations. The example reveals that variations in the implementations of a single dimension of the design space can have a considerable impact on the PM output (here: *Measure Functionality*).

6.7 T6—Compare Few PMs over Time

Two aspects add to the complexity of T6. On the one hand, we aim at observing a labeling process over time. On the other hand, this observation focuses on the comparison of several PMs. We choose a grid-based layout for the visual comparison of PMs over time. The changes of PMs across the labeling process are visualized from left to right, whereas the output of different PMs can be compared vertically (multiple lines). Figure 22 shows the result of such a labeling process. In this scenario, we have decided that a single labeling process was performed, and, in every iteration, all PMs show the same state (within every column all PMs work with the same unlabeled dataset and training set). The decisive benefit of this approach is the ability to compare both aspects *temporal developments* and *PM differences* across a unified labeling process.

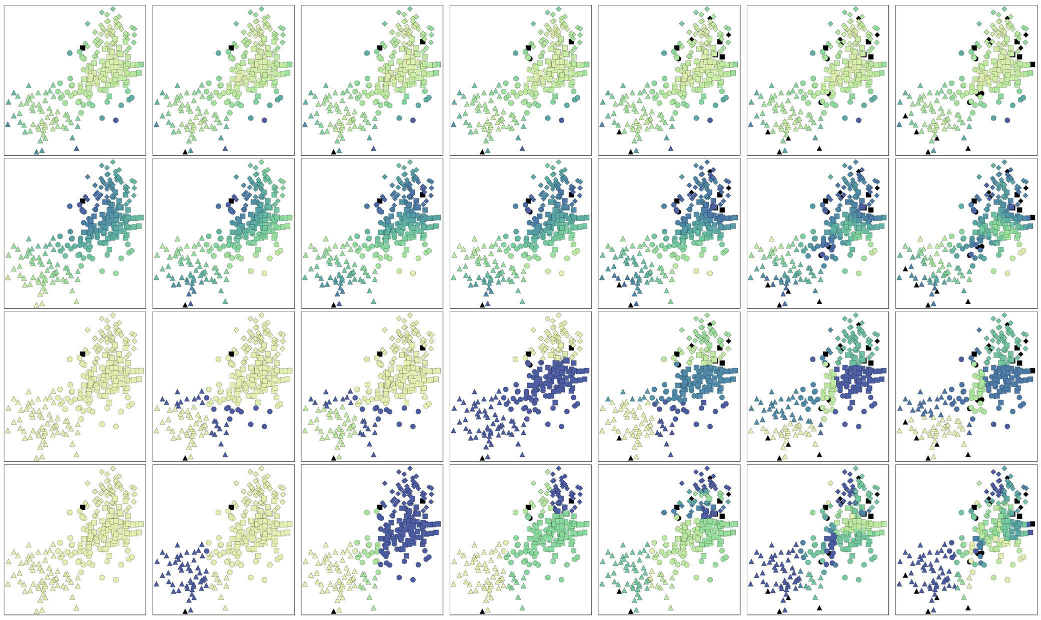


Fig. 22. Observation of four different PMs (one per row) within a unified labeling process (left to right columns) (T6) for a Default Dataset. Iterations 1, 2, 3, 4, 10, 15, and 20 are shown. Row 1: PM based on *density*, showing smallest nearest neighbor distances for iterations. Row 2: PM based on *coverage*, showing distances of unlabeled data to the training set. Row 3: PM based on *relevance*, showing the lowest entropy of class predictions for iterations. Row 4: PM based on *uncertainty*, showing least significant confidences. While density information (row 1) is very useful in early labeling iterations [24, 118], the density PM remains constant over time. The reason is simple: the density is calculated on both unlabeled and labeled data, thus there is no change throughout the labeling process. In contrast, the PM based on coverage (row 2) is sensitive to the distribution of training data, and highlights unlabeled instances farthest away from training data. The PM based on relevance (row 3) confirms already existing class structures and highlights instances near already labeled regions. In contrast, the PM based on uncertainty (row 4) highlights regions where an underlying classifier is most uncertain.

6.8 T7—Overview across All PMs

Figure 23 demonstrates how we used the PM Gallery View of the explainer to gain an overview of the 605 PM implementations we are currently using. The gallery shows the considerable differences of the PM outputs. Another pattern that can be observed frequently is the complete inversion of PM outputs (colors appear to be inverse), which can be explained by the *Measure Valence* dimension when applied in either way "high is good" and "low is good." An interesting subgroup of six PMs is selected (red bold outlines) showing PMs that are particularly similar: all PMs belong to the *uncertainty* property (details described in the figure caption). A drill-down from the analysis of all PMs to this subgroup of PMs is shown in Figure 25 where the PM Tree View is used (detailed analysis in the caption). The tree representation clearly underlines that it can be used to validate the alignment of PM sub-groups with dimensions of the taxonomy.

The PM Tree View is the second interface that allows gaining an overview of the 605 PM implementations. Using Figure 24 as an example, we structure all PM implementations by the property they quantify. At a glance, it can be identified that many PM implementations within properties are similar, whereas PMs between properties behave differently. This also builds a baseline for downstream analyses like T8, the comparison between designated subgroups of PMs.

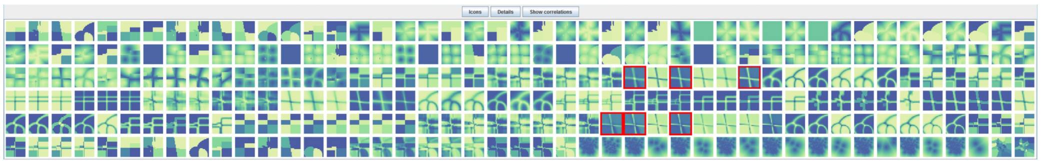


Fig. 23. PM Gallery View providing an overview of PMs (T7) (lower half is truncated). The grid-based interface shows icons of PMs using the Data Space Coloring mode. In the example, 6 PMs are selected that are very similar. Analysis of the selected subset reveals that all PMs use the probabilistic output of the same classifier (Multilayer Perceptron). All PMs address the same property (uncertainty), used in AL for uncertainty sampling strategies. However, The implementations of the six PMs differ. The aggregation functionalities are Entropy, least significant confidence, and the Gini index. The three PMs in the higher row are instance neighbor-based, i.e., they also take the information from neighbors into account. As a result, the PM output is more smooth and robust compared to the output of the three PMs based instance-only implementations.

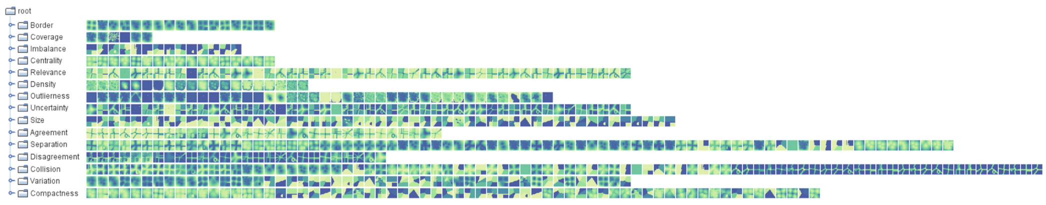


Fig. 24. Overview of PMs we implemented for the visual analytics explainer tool. PMs reveal data or model output characteristics and can, e.g., be used to support data labeling tasks. Small icons show the output of every PM, all calculated for the same dataset. The explainer reveals the variety of different characteristics, which at a coarsest level can be subdivided into 15 primary properties. The tree visualization of the explainer is used to align PM implementations to the properties they explain.

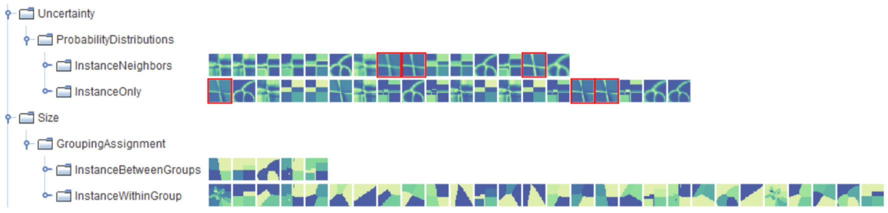


Fig. 25. PM Tree View expanded to the local substructures uncertainty and size. We re-identify the six PMs from Figure 23 that have previously been selected. Now, it is evident that these particularly similar PM outputs all belong to the same property (uncertainty). Similarly, the differentiation between instance neighbors and instance only can easily be assessed using the view. These analyses help to validate PM implementations, as well as the validation of the descriptive power of the taxonomy. To give another example for the taxonomy validity, all PMs within the uncertainty property are fairly similar, but differ considerably to the group of PMs aligned with the size property.

6.9 T8—Compare between Designated Subgroups of PMs

In line with the findings we had in the overview task (T7) previously, we further focus on the smaller subgroups of PMs. Of particular interest are commonalities and differences of PM subgroups. As an ultimate goal, we use T8 to validate the taxonomy by aligning subgroups of PMs with the taxonomy. For that purpose, we use the PM Tree View of the explainer (cf. Section 5.4) to analyze the sub-division of PMs into groups according to the taxonomy.

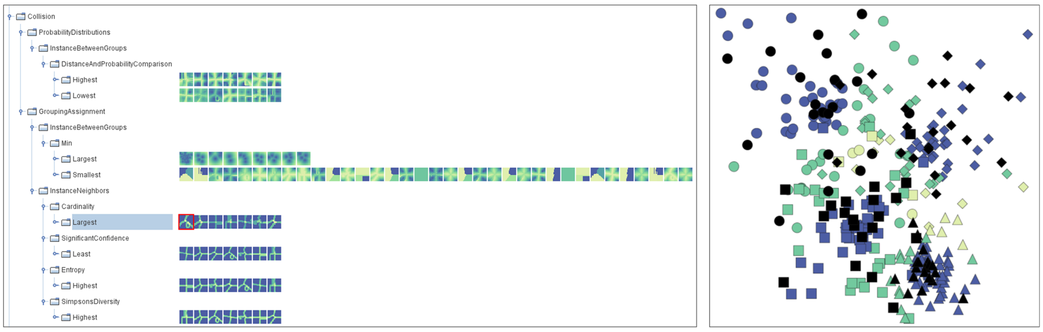


Fig. 26. Left: Tree visualization used for the visual comparison of designated subgroups of PMs of the *collision* property (T8). Right: Detailed analysis of a selected PM. Overall, the tree reveals that eight subgroups in the design space contain implementations for the *collision* property. The PMs in the upper part of the tree use a probabilistic classifier output and assess collisions between groups. The two subgroups below use the groupings of clustering algorithms to facilitate between-group assessment. At the bottom, four similar PM subgroups stand out, following yet another approach: The group assignments of classifiers are used in combination with instance neighbor comparisons, to identify local collisions of class predictions for this Default Dataset. The selected PM (largest vote cardinality) highlights the region in the dataset where neighbor instances have different class assignments. In the PM Detail View at the right, two regions with bright colors stand out: here, neighboring instances are classified most differently, with particularly severe class collisions.

One example case is the detailed analysis of one property together with all PM implementations supporting this property. We decide to focus on the *collision* property, as it contains the largest number of PM implementations (cf. Figure 24). While Figures 8 and 9 show the alignment of properties with the design space, we use the explainer to align all PMs of the *collision* property with designated subgroups. Figure 26 shows how the PMs align across the subgroups.

7 DISCUSSION

Our iterative refinement process achieved a characterization of PMs and the PM taxonomy using the explainer that was co-developed with them. We now discuss additional insights that arose from that formative design process and our summative analysis of them, discuss aspects that we did not formalize explicitly or add to the taxonomy, and outline ideas for future work.

7.1 Role of Datasets

The design space for PMs has no restrictions or limitations with respect to the dimensionality of the applied dataset. However, the visual explainer uses a 2D representation of data, which requires dimensionality reduction for multivariate data. While using multiple dimensionality reduction techniques visualized side-by-side may help mitigating problems stemming from a singleton algorithm building upon a single optimization criterion, one overarching question remains: Can the error introduced by dimensionality reduction be quantified for visual-interactive labeling tasks? Such an assessment would help to make visual-interactive labeling interfaces more effective.

7.2 Role of ML Models

The analyses about the explanation of different classifier characteristics (cf. T2 in Section 6.3) revealed that the internal structures of classifiers differ considerably. Thereby PMs played an important role as they showed to be a useful means to reveal internal classifier characteristics (e.g., areas of confidence and decision boundaries). Overall, there is a dependency between the selection of PMs, ML models and the achieved output performance, which is worth a further investigation.

Interesting research questions are

- Which PMs are most promising in combination with which ML model?
- Is it possible to infer good model choices, given the idea to implement a PM for some property?
- Which role can visualization play to support appropriate model choices?

7.3 Potential Extensions beyond Model Outputs to Models

In the construction of the taxonomy and the design space, we have tried to focus on commonalities between the different domains and perspectives rather than on domain-specific aspects. This perspective has led us to some model-related abstractions. One limitation of our approach is that PMs use the model output as an abstraction, which does not allow access to any internal information from an ML models. However, some AL strategies require more internal information from the classifier. Variants of AL-based *uncertainty* sampling strategies are one example, often measuring the distance of the instance in feature space from the closest class boundary. Besides, *error reduction* schemes [117] are the class of AL strategies that particularly make use of classifier internals. These strategies try to select instances that either would change the model most during training (expected model change) or that would reduce the expected generalization error of the model more (expected error reduction) [117]. A second limitation is the focus on the output on one particular type of model output per PM. While we support using ensembles of the same model class, the taxonomy currently does not envisage to take model outputs from different model classes, such as results from a classification and a regression model. If such complexity is needed, then the definition of PMs and the constraints in the design space could be adapted.

To cope with limitations for model-centric AL strategies, we outline four possible extensions that may be characterized, abstracted, and integrated into the taxonomy. An obvious approach would be to extend the *Model Output* dimension of our taxonomy through a *PM-Model Interplay* dimension, that is able to cope with the following.

- **Model data:** providing additional (specific) information about ML model characteristics, such as decision boundaries.
- **Model usage:** providing access to trained ML models to allow PMs gaining specific and model-internal information
- **Model training:** allowing PMs to modify or re-train ML models, e.g., to assess expected error changes
- **Multi models:** allowing PMs to access more than one type of model class.

7.4 Applications beyond Data Labeling

We have created the PM taxonomy with a focus on data labeling tasks, but the definition of PMs as well as the design space allows the usage of PMs for other applications. It would be conceivable to compare PMs with quality metrics as they exist in the visualization community, for example to assess cluster separability [114]. Both approaches have in common that output values are provided for single instances. An interesting difference is the application of PMs in early states of data science workflows (data, model), while quality metrics in the visualization community primarily assess late stages, such as the visual space after the view transformation in a visualization pipeline [36].

Another application context is provided with eye tracking technology and the wealth of existing eye tracking data. Similarly to our qualitative visual data analyses in Section 6, the visual comparison of these patterns with the output of different PMs is feasible, possibly leading to meaningful relations.

Finally, it would be interesting to have a stronger focus on regression model rather than classification models. The classification-based focus is due to the the categorical data labeling task. The analysis of PMs for quantitative supervised ML tasks is a natural area for future work.

7.5 Combination of Property Measures

The formalization of PMs, the taxonomy and the explainer is a stepping stone toward the synthesis of instance selection strategies with better performance than existing strategies. The foundation provided in this article will support strategy synthesis based on carefully chosen combinations of PMs. Given the insights gained with our analyses (cf. Section 6), it must also be assumed that such combinations of meaningful properties may even need to change during the labeling process that would mean a paradigm change, given that existing instance selection strategies do not change their "ingredients" during a labeling process. We hope that the synthesis of such instance selection strategies will be the subject of fruitful future work.

7.6 Role of Dimensionality Reduction

We use dimensionality reduction as a standard approach to map multivariate data into 2D, i.e., into the visual space represented with scatterplots. While dimensionality reduction is not the core focus of this work, we still want to highlight two discussion points. First, dimensionality reduction always introduces errors when multivariate data is mapped into 2D and there is a huge space of possibilities to quantify these errors [50] at different granularities (dataset-global or instance-based) and in the way how errors are conveyed to the user (visually or non-visually). Depending on the scope, adding visual quality assessment for dimensionality reduction techniques can be a useful extension of the explainer. Second, our Data Space Coloring mode (cf. Section 5.2) requires an inverse projection from the visual space back into the original data space. Again, we recall that we do support Data Space Coloring for 2D datasets and for multivariate datasets with linear dimensionality reduction techniques like PCA, but do not yet support non-linear techniques for this mode. Solutions to inverse projections for also non-linear dimensionality reduction techniques have been presented recently [51, 109] and may be added to the explainer in future.

7.7 Interaction Scalability

The explainer provides user interaction to support data manipulation and what-if-analyses. While these types of analyses have proven to be quite useful especially for T3 (cf. Section 6.4), they have limitations regarding scalability. Especially the drag-and-drop interaction of instances in the visual space is computationally expensive. The motion of instances requires an inverse mapping from the visual space to the data space as well as the recalculation of ML models attached to the focused PM. In most cases when we conducted what-if-analyses, we used Demo Datasets, as opposed to large datasets that are not real-time capable.

8 CONCLUSION

We presented, formalized, and characterized the idea of a PM as the lowest-level building block to synthesize instance selection strategies. PMs receive an instance as input and provide a characteristics about the involved data or the output of a machine learning model, or both. Instance selection strategies, commonly used in the data labeling process in machine learning and in visual analytics, can now be characterized by their properties as their common level of description and be quantified with PMs. With our PM taxonony, we have defined a design space for PMs with the descriptive power to differentiate between PMs and the generative power to reveal designs for novel PMs. The process of defining and characterizing PMs and the PM taxonomy was accompanied by the development of a VA explainer tool that was useful to assess data and model characteristics revealed by PMs, gain insight into the interactive data labeling process, and validate decisions made regarding PM implementation and taxonomy design. For summative validation purposes, we presented a series of qualitative visual data analyses along eight carefully designed analysis tasks. With the proposal of PMs, the PM taxonomy, and the explainer using PMs, we have made one step further toward the creation of novel instance selection strategies that may exploit the full power of data and model properties.

REFERENCES

[1] Mostafa M. Abbas, Michaël Aupetit, Michael Sedlmair, and Halima Bensmail. 2019. ClustMe: A visual quality measure for ranking monochrome scatterplots based on cluster patterns. *Comput. Graph. Forum* 38, 3 (2019), 225–236. https://doi.org/10.1111/cgf.13684

[2] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160.

[3] Charu C. Aggarwal. 2015. Outlier analysis. In *Data Mining*. Springer, 237–263.

[4] E. Alexander and M. Gleicher. 2016. Task-driven comparison of topic models. *IEEE Trans. Vis. Comput. Graph.* 22, 1 (2016), 320–329. https://doi.org/10.1109/TVCG.2015.2467618

[5] B. Alsallakh, A. Hanbury, H. Hauser, S. Miksch, and A. Rauber. 2014. Visual methods for analyzing probabilistic classification data. *IEEE Trans. Vis. Comput. Graph.* 20, 12 (2014), 1703–1712. https://doi.org/10.1109/TVCG.2014.2346660

[6] Saleema Amershi, Max Chickering, Steven Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. ModelTracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'15)*.

[7] D. Arthur and S. Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*. 1027–1035.

[8] Vijay Arya, Rachel K. E. Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Q. Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. 2019. One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques. arXiv:1909.03012. Retrieved from https://arxiv.org/abs/1909.03012.

[9] Josh Attenberg and Foster Provost. 2011. Inactive Learning? Difficulties employing active learning in practice. *SIGKDD Explor. Newsl.* 12, 2 (2011), 36–41. https://doi.org/10.1145/1964897.1964906

[10] M. Aupetit and M. Sedlmair. 2016. SepMe: 2002 new visual separation measures. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis'16)*. 1–8. https://doi.org/10.1109/PACIFICVIS.2016.7465244

[11] Stéphane Ayache and Georges Quénot. 2007. Evaluation of active learning strategies for video indexing. *Sign. Process.: Image Commun.* 22, 7-8 (2007), 692–704.

[12] Robert Babuška. 2012. *Fuzzy Modeling for Control*. Vol. 12. Springer Science & Business Media.

[13] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.* 6, 1 (Jun. 2004), 20–29. https://doi.org/10.1145/1007730.1007735

[14] M. Beaudouin-Lafon. 2004. Designing interaction, not interfaces. In *Proceedings of the Advanced Visual Interfaces (AVI'04)*. 15–22. http://dx.doi.org/10.1145/989863.989865

[15] M. Behrisch, B. Bach, M. Hund, M. Delz, L. Von Rüden, J. Fekete, and T. Schreck. 2017. Magnostics: Image-based search of interesting matrix views for guided network exploration. *IEEE Trans. Vis. Comput. Graph.* 23, 1 (2017), 31–40. https://doi.org/10.1109/TVCG.2016.2598467

[16] M. Behrisch, F. Korkmaz, Lin Shao, and T. Schreck. 2014. Feedback-driven interactive exploration of large multidimensional data supported by visual classifier. In *Proceedings of the IEEE Visual Analytics Science and Technology (VAST'14)*. 43–52. https://doi.org/10.1109/VAST.2014.7042480

[17] B. C. Benato, A. C. Telea, and A. X. Falcão. 2018. Semi-supervised learning with interactive label propagation guided by feature space projections. In *Proceedings of the Graphics, Patterns and Images (SIBGRAPI'18)*. 392–399. https://doi.org/10.1109/SIBGRAPI.2018.00057

[18] Jürgen Bernard, Marco Hutter, Markus Lehmann, Martin Müller, Matthias Zeppelzauer, and Michael Sedlmair. 2018. Learning from the best—Visual analysis of a quasi-optimal data labeling strategy. In *Proceedings of the Conference on Visualization (EuroVis'18)*. Eurographics. https://doi.org/10.2312/eurovisshort.20181085

[19] Jürgen Bernard, Marco Hutter, Christian Ritter, Markus Lehmann, Michael Sedlmair, and Matthias Zeppelzauer. 2019. Visual analysis of degree-of-interest functions to support selection strategies for instance labeling. In *Proceedings of the EuroVis Workshop on Visual Analytics (EuroVA'19)*. The Eurographics Association. https://doi.org/10.2312/eurova.20191116

[20] J. Bernard, M. Hutter, M. Zeppelzauer, D. Fellner, and M. Sedlmair. 2018. Comparing visual-interactive labeling with active learning: An experimental study. *IEEE Trans. Vis. Comput. Graph.* 24, 1 (2018), 298–308. https://doi.org/10.1109/TVCG.2017.2744818

[21] Jürgen Bernard, Marco Hutter, Matthias Zeppelzauer, Michael Sedlmair, and Tamara Munzner. 2020. SepEx: Visual analysis of class separation measures. In *Proceedings of the EuroVis Workshop on Visual Analytics (EuroVA'20).* The Eurographics Association. https://doi.org/10.2312/eurova.20201079

[22] Jürgen Bernard, David Sessler, Andreas Bannach, Thorsten May, and Jörn Kohlhammer. 2015. A visual active learning system for the assessment of patient well-being in prostate cancer research. In *Proceedings of the IEEE VIS WS on Visual Analytics in Healthcare (VAHC'15).* ACM, Article 1, 8 pages. https://doi.org/10.1145/2836034.2836035

[23] Jürgen Bernard, David Sessler, Tobias Ruppert, James Davey, Arjan Kuijper, and Jörn Kohlhammer. 2014. User-based visual-interactive similarity definition for mixed data objects-concept and first implementation. *J. WSCG* 22 (2014), 329–338.

[24] Jürgen Bernard, Matthias Zeppelzauer, Markus Lehmann, Martin Müller, and Michael Sedlmair. 2018. Towards user-centered active learning algorithms. *Comput. Graph. For.* (2018). https://doi.org/10.1111/cgf.13406

[25] Jürgen Bernard, Matthias Zeppelzauer, Michael Sedlmair, and Wolfgang Aigner. 2018. VIAL: A unified process for visual interactive labeling. *Vis. Comput.* 34, 9 (2018), 1189–1207. https://doi.org/10.1007/s00371-018-1500-3

[26] David M. Blei. 2012. Probabilistic topic models. *Commun. ACM* 55, 4 (Apr. 2012), 77–84. https://doi.org/10.1145/2133806.2133826

[27] Phillip Bonacich. 1987. Power and centrality: A family of measures. *Am. J. Sociol.* 92, 5 (1987), 1170–1182.

[28] Leo Breiman. 2001. Random forests. *Mach. Learn.* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[29] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying density-based local outliers. *SIGMOD Rec.* 29, 2 (May 2000), 93–104. https://doi.org/10.1145/335191.335388

[30] Eli T. Brown, Jingjing Liu, Carla E. Brodley, and Remco Chang. 2012. Dis-function: Learning distance functions interactively. In *Proceedings of the IEEE Visual Analytics Science and Technology (VAST'12).* IEEE, 83–92.

[31] Ángel Alexander Cabrera, Fred Hohman, Jason Lin, and Duen Horng Chau. 2018. Interactive classification for deep learning interpretation. arXiv:1806.05660. Retrieved from https://arxiv.org/abs/1806.05660.

[32] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41, 3, Article 15 (2009), 58 pages. https://doi.org/10.1145/1541880.1541882

[33] Mohammad Chegini, Jürgen Bernard, Philip Berger, Alexei Sourin, Keith Andrews, and Tobias Schreck. 2019. Interactive labelling of a multivariate dataset for supervised machine learning using linked visualisations, clustering, and active learning. *Vis. Inf.* 3, 1 (2019), 9–17. https://doi.org/10.1016/j.visinf.2019.03.002

[34] Mohammad Chegini, Jürgen Bernard, Lin Shao, Alexei Sourin, Keith Andrews, and Tobias Schreck. 2019. mVis in the Wild: Pre-Study of an interactive visual machine learning system for labelling. In *Proceedings of the IEEE VIS Workshop on Evaluation of Interactive Visual Machine Learning Systems.*

[35] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems.* 2172–2180.

[36] E. H. Chi. 2000. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'00).* 69–75. https://doi.org/10.1109/INFVIS.2000.885092

[37] John G. Cleary and Leonard E. Trigg. 1995. K*: An instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine Learning.* 108–114.

[38] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Mach. Learn.* 20, 3 (1995), 273–297.

[39] Nathan Altshiller Court. 1960. Notes on the centroid. *Math. Teach.* 53, 1 (1960), 33–35.

[40] Piotr Dabkowski and Yarin Gal. 2017. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems.* 6967–6976.

[41] Tuan Nhon Dang, Anushka Anand, and Leland Wilkinson. 2013. Timeseer: Scagnostics for high-dimensional time series. *IEEE Trans. Vis. Comput. Graph.* 19, 3 (2013), 470–483. https://doi.org/10.1109/TVCG.2012.128

[42] Tuan Nhon Dang and Leland Wilkinson. 2014. Transforming scagnostics to reveal hidden features. *IEEE Trans. Vis. Comput. Graph.* 20, 12 (2014), 1624–1632. https://doi.org/10.1109/TVCG.2014.2346572

[43] A. Dasgupta and R. Kosara. 2010. Pargnostics: Screen-space metrics for parallel coordinates. *IEEE Trans. Vis. Comput. Graph.* 16, 6 (2010), 1017–1026. https://doi.org/10.1109/TVCG.2010.184

[44] A. Dasgupta, H. Wang, N. O'Brien, and S. Burrows. 2019. Separating the wheat from the chaff: Comparative visual cues for transparent diagnostics of competing models. *IEEE Trans. Vis. Comput. Graph.* (2019), 1–1. https://doi.org/10.1109/TVCG.2019.2934540

[45] D. L. Davies and D. W. Bouldin. 1979. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.* 1, 2 (Apr. 1979), 224–227. https://doi.org/10.1109/TPAMI.1979.4766909

[46] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc. Ser. B* 39, 1 (1977), 1–38. http://www.jstor.org/stable/2984875.

[47] Frederik L. Dennig, Tom Polk, Zudi Lin, Tobias Schreck, Hanspeter Pfister, and Michael Behrisch. 2019. FDive: Learning relevance models using pattern-based similarity measures. arXiv:1907.12489. Retrieved from http://arxiv.org/abs/1907.12489.

[48] Richard O. Duda, Peter E. Hart, and David G. Stork. 2012. *Pattern Classification.* John Wiley & Sons.

[49] Joseph C. Dunn. 1974. Well-separated clusters and optimal fuzzy partitions. *Cybernet. Syst.* 4, 1 (1974), 95–104.

[50] Mateus Espadoto, Rafael M. Martins, Andreas Kerren, Nina S. T. Hirata, and Alexandru Cristian Telea. 2019. Towards a quantitative survey of dimension reduction techniques. *IEEE Trans. Vis. Comput. Graph.* (2019).

[51] Mateus Espadoto, Francisco Caio Maia Rodrigues, Nina S. T. Hirata, Roberto Hirata Jr., and Alexandru C. Telea. 2019. Deep learning inverse multidimensional projections. In *Proceedings of the EuroVis Workshop on Visual Analytics (EuroVA'19).* The Eurographics Association. https://doi.org/10.2312/eurova.20191118

[52] Brendan J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science* 315, 5814 (2007), 972–976. https://doi.org/10.1126/science.1136800 arXiv:https://science.sciencemag.org/content/315/5814/972.full.pdf.

[53] Nir Friedman, Dan Geiger, and Moises Goldszmidt. 1997. Bayesian network classifiers. *Mach. Learn.* 29, 2 (01 Nov. 1997), 131–163. https://doi.org/10.1023/A:1007465528199

[54] Yifan Fu, Xingquan Zhu, and Bin Li. 2013. A survey on instance selection for active learning. *Knowl. Inf. Syst.* 35, 2 (01 May 2013), 249–283. https://doi.org/10.1007/s10115-012-0507-8

[55] B. Fuglede and F. Topsoe. 2004. Jensen-Shannon divergence and Hilbert space embedding. In *Proceedings of the International Symposium on Information Theory (ISIT'04).* 31. https://doi.org/10.1109/ISIT.2004.1365067

[56] Corrado Gini. 1912. Variabilità e mutabilità. Reprinted in *Memorie di metodologica statistica*, E. Pizetti and T. Salvemini. Libreria Eredi Virgilio Veschi, Rome (1912).

[57] Philippe H. Gosselin and Matthieu Cord. 2004. A comparison of active classification methods for content-based image retrieval. In *Proceedings of the International Workshop on Computer Vision Meets Databases.* ACM, 51–58.

[58] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. 2002. Clustering validity checking methods: Part II. *SIGMOD Rec.* 31, 3 (2002), 19–27. https://doi.org/10.1145/601858.601862

[59] Jiawei Han, Jian Pei, and Micheline Kamber. 2012. *Data Mining: Concepts and Techniques.* Elsevier. https://doi.org/10.1016/C2009-0-61819-5

[60] F. Heimerl, S. Koch, H. Bosch, and T. Ertl. 2012. Visual classifier training for text document retrieval. *IEEE Trans. Vis. Comput. Graph.* 18, 12 (2012), 2839–2848. https://doi.org/10.1109/TVCG.2012.277

[61] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating visual explanations. In *Proceedings of the European Conference on Computer Vision.* Springer, 3–19.

[62] Andreas Hinterreiter, Peter Ruch, Holger Stitz, Martin Ennemoser, Jürgen Bernard, Hendrik Strobelt, and Marc Streit. 2019. ConfusionFlow: A model-agnostic visualization for temporal analysis of classifier confusion. arXiv:cs.LG/1910.00969

[63] Dorit S. Hochbaum and David B. Shmoys. 1985. A best possible heuristic for the k-center problem. *Math. Operat. Res.* 10, 2 (1985), 180–184.

[64] Benjamin Höferlin, Rudolf Netzel, Markus Höferlin, Daniel Weiskopf, and Gunther Heidemann. 2012. Inter-active learning of ad-hoc classifiers for video visual analytics. In *Proceedings of the Visual Analytics Science and Technology (VAST'12).* IEEE, 23–32. https://doi.org/10.1109/VAST.2012.6400492

[65] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. 2019. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Trans. Vis. Comput. Graph.* 25, 8 (2019), 2674–2693. https://doi.org/10.1109/TVCG.2018.2843369

[66] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. 2006. Large-scale text categorization by batch mode active learning. In *Proceedings of the Annual Conference on the World Wide Web.* ACM, 633–642. https://doi.org/10.1145/1135777.1135870

[67] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Netw.* 2, 5 (1989), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

[68] Lulu Huang, Stan Matwin, Eder J. de Carvalho, and Rosane Minghim. 2017. Active learning with visualization for text data. In *Proceedings of the ACM WS on Exploratory Search and Interactive Data Analytics (ESIDA'17).* ACM, 69–74. https://doi.org/10.1145/3038462.3038469

[69] Anil K. Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern Recogn. Lett.* 31, 8 (2010), 651–666.

[70] Nathalie Japkowicz and Shaju Stephen. 2002. The class imbalance problem: A systematic study. *Intell. Data Anal.* 6, 5 (Oct. 2002), 429–449. http://dl.acm.org/citation.cfm?id=1293951.1293954.

[71] Johan Ludwig William Valdemar Jensen, et al. 1906. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Math.* 30 (1906), 175–193.

[72] I. T. Jolliffe. 2002. *Principal Component Analysis* (3rd ed.). Springer.

[73] Marius Kaminskas and Derek Bridge. 2017. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Trans. Interact. Intell. Syst.* 7, 1 (2017), 2:1–2:42. https://doi.org/10.1145/2926720

[74] Been Kim, Rajiv Khanna, and Oluwasanmi O. Koyejo. 2016. Examples are not enough, learn to criticize! Criticism for Interpretability. In *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2280–2288.

[75] Bongjun Kim and Bryan Pardo. 2018. A Human-in-the-Loop system for sound event detection and annotation. *ACM Trans. Interact. Intell. Syst.* 8, 2 (2018), 13:1–13:23. https://doi.org/10.1145/3214366

[76] Edwin M. Knorr and Raymond T. Ng. 1998. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the Conference on Very Large Data Bases (VLDB'98)*. Morgan Kaufmann, 392–403.

[77] Andrey Kolmogorov. 1933. Sulla determinazione empirica di una legge di distribuzione. *G. Ist. Ital. Attuari* 4 (1933), 83–91.

[78] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. 2017. Learning active learning from data. In *Advances in Neural Information Processing Systems*. 4226–4236.

[79] P. Köthur, M. Sips, H. Dobslaw, and D. Dransch. 2014. Visual analytics for comparison of ocean model output with reference data: Detecting and analyzing geophysical processes using clustering ensembles. *IEEE Trans. Vis. Comput. Graph.* 20, 12 (2014), 1893–1902. https://doi.org/10.1109/TVCG.2014.2346751

[80] Daniel Kottke, Adrian Calma, Denis Huseljic, Georg Krempl, and Bernhard Sick. 2017. Challenges of reliable, realistic and comparable active learning evaluation. In *Proceedings of the Workshop and Tutorial on Interactive Adaptive Learning*. 2–14.

[81] Josua Krause, Adam Perer, and Kenney Ng. 2016. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'16)*. ACM, New York, NY, 5686–5697. https://doi.org/10.1145/2858036.2858529

[82] Hans-Peter Kriegel, Matthias S. hubert, and Arthur Zimek. 2008. Angle-based outlier detection in high-dimensional data. In *Proceedings of the SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 444–452.

[83] Joseph B. Kruskal. 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* 29, 1 (1964), 1–27.

[84] Kostiantyn Kucher, Carita Paradis, Magnus Sahlgren, and Andreas Kerren. 2017. Active learning and visual analytics for stance classification with ALVA. *ACM Trans. Interact. Intell. Syst.* 7, 3 (2017), 14:1–14:31. https://doi.org/10.1145/3132169

[85] S. Kullback, R. A. Leibler, et al. 1951. On information and sufficiency. *Ann. Math. Stat.* 22, 1 (1951), 79–86.

[86] Niels Landwehr, Mark Hall, and Eibe Frank. 2005. Logistic model trees. *Mach. Learn.* 59, 1–2 (2005), 161–205. https://doi.org/10.1007/s10994-005-0466-3

[87] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[88] Dirk J. Lehmann, Fritz Kemmler, Tatsiana Zhyhalava, Marco Kirschke, and Holger Theisel. 2015. Visualnostics: Visual guidance pictograms for analyzing projections of high-dimensional data. *Comput. Graph. Forum* 34, 3 (2015), 291–300. https://doi.org/10.1111/cgf.12641

[89] Qimai Li, Xiao-Ming Wu, and Zhichao Guan. 2019. Generalized label propagation methods for semi-supervised learning. arXiv:1901.09993. Retrieved from https://arxiv.org/abs/1901.09993.

[90] F. C. M. Rodrigues, R. Hirata, and A. C. Telea. 2018. Image-based visualization of classifier decision boundaries. In *Proceedings of the IEEE Conference on Graphics, Patterns and Images (SIBGRAPI'18)*. 353–360. https://doi.org/10.1109/SIBGRAPI.2018.00052

[91] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 9, (Nov. 2008), 2579–2605.

[92] Naoki Abe Hiroshi Mamitsuka. 1998. Query learning strategies using boosting and bagging. In *Proceedings of the International Conference on Machine Learning (ICML'98)*, Vol. 1. Morgan Kaufmann.

[93] José Matute, Alexandru C. Telea, and Lars Linsen. 2018. Skeleton-based scagnostics. *IEEE Trans. Vis. Comput. Graph.* 24, 1 (2018), 542–552.

[94] Andrew McCallum and Kamal Nigam. 1998. Employing EM and pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning (ICML'98)*. Morgan Kaufmann, San Francisco, CA, 350–358.

[95] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*. Citeseer, 169–178.

[96] Dalibor Mitrović, Matthias Zeppelzauer, and Christian Breiteneder. 2010. Features for content-based audio retrieval. In *Advances in Computers*. Vol. 78. Elsevier, 71–150.

[97] Rana Momtaz, Nesma Mohssen, and Mohammad A. Gowayyed. 2013. DWOF: A robust density-based outlier detection approach. In *Pattern Recognition and Image Analysis*. Springer, 517–525.

[98] Fionn Murtagh. 1983. A survey of recent advances in hierarchical clustering algorithms. *Comput. J.* 26, 4 (1983), 354–359. https://doi.org/10.1093/comjnl/26.4.354

[99] Fredrik Olsson. 2008. *Bootstrapping Named Entity Annotation by Means of Active Machine Learning: A Method for Creating Corpora.* Ph.D. Dissertation, SICS.

[100] Fredrik Olsson. 2009. *A Literature Survey of Active Machine Learning in the Context of Natural Language Processing.* Technical Report. Swedish Institute of Computer Science.

[101] Fernando V. Paulovich, Maria Cristina F. Oliveira, and Rosane Minghim. 2007. The projection explorer: A flexible tool for projection-based multidimensional visualization. In *Proceedings of the Symposium on Computer Graphics and Image Processing (SIBGRAPI'07).* IEEE Computer Society, 27–36. https://doi.org/10.1109/SIBGRAPI.2007.21

[102] Dan Pelleg and Andrew W. Moore. 2000. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proceedings of the Conference on Machine Learning.* Morgan Kaufmann, 727–734.

[103] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. F. Lelieveldt, E. Eisemann, and A. Vilanova. 2018. DeepEyes: Progressive visual analytics for designing deep neural networks. *IEEE Trans. Vis. Comput. Graph.* 24, 1 (2018), 98–108. https://doi.org/10.1109/TVCG.2017.2744358

[104] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, and Hong-Jiang Zhang. 2009. Two-dimensional multilabel active learning with an efficient online adaptation model for image classification. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 10 (2009), 1880–1897. https://doi.org/10.1109/TPAMI.2008.218

[105] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the Conference on Management of Data (SIGMOD'00).* ACM, 427–438. https://doi.org/10.1145/342009.335437

[106] Paulo E. Rauber, Alexandre X. Falcão, and Alexandru C. Telea. 2018. Projections as visual aids for classification system design. *Inf. Vis.* 17, 4 (2018), 282–305. https://doi.org/10.1177/1473871617713337

[107] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. 2017. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Trans. Vis. Comput. Graph.* 23, 1 (Jan. 2017), 61–70. https://doi.org/10.1109/TVCG.2016.2598828

[108] R. A. Rensink and Gideon Baldridge. 2010. The perception of correlation in scatterplots. *Comput. Graph. Forum* 29, 3 (2010), 1203–1210.

[109] Francisco Rodrigues, Mateus Espadoto, Roberto Hirata, and Alexandru C. Telea. 2019. Constructing and visualizing high-quality classifier decision boundary maps. *Information* 10, 9 (2019), 280.

[110] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20 (1987), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

[111] S. Santini and R. Jain. 1999. Similarity measures. *IEEE Trans. Pattern Anal. Mach. Intell.* 21, 9 (1999), 871–883. https://doi.org/10.1109/34.790428

[112] Jorn Schneidewind, Mike Sips, and Daniel A. Keim. 2006. Pixnostics: Towards measuring the value of visualization. In *Proceedings of the IEEE Visual Analytics Science and Technology (VAST'06).* 199–206.

[113] M. Sedlmair and M. Aupetit. 2015. Data-driven evaluation of visual quality measures. *Comput. Graph. Forum* 34, 3 (2015), 201–210. https://doi.org/10.1111/cgf.12632

[114] M. Sedlmair, A. Tatu, T. Munzner, and M. Tory. 2012. A taxonomy of visual cluster separation factors. *Comput. Graph. Forum* 31, 3pt4 (2012), 1335–1344. https://doi.org/10.1111/j.1467-8659.2012.03125.x

[115] C. Seifert and M. Granitzer. 2010. User-based active learning. In *Proceedings of the IEEE Conference on Data Mining Workshops (ICDMW'10).* 418–425. https://doi.org/10.1109/ICDMW.2010.181

[116] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2016. Grad-CAM: Why did you say that? arXiv:1611.07450. Retrieved from https://arxiv.org/abs/1611.07450.

[117] Burr Settles. 2009. *Active Learning Literature Survey.* Technical Report 1648. University of Wisconsin–Madison.

[118] Burr Settles. 2012. Active learning. *Synth. Lect. Artif. Intell. Mach. Learn.* 6, 1 (2012), 1–114.

[119] Burr Settles and Mark Craven. 2008. An Analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Annual Conference on Empirical Methods in Natural Language Processing (EMNLP'08).* Computational Linguistics, 1070–1079.

[120] Burr Settles, Mark Craven, and Soumya Ray. 2008. Multiple-instance active learning. In *Advances in Neural Information Processing Systems.* 1289–1296.

[121] H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proceedings of the Workshop on Computer Learning Theory (COLT'92).* ACM, 287–294. https://doi.org/10.1145/130385.130417

[122] C. E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Techn. J.* 27, 3 (1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[123] Edward H. Simpson. 1949. Measurement of diversity. *Nature* 163, 4148 (1949), 688.

[124] Mike Sips, Boris Neubert, John P. Lewis, and Pat Hanrahan. 2009. Selecting good views of high-dimensional data using class consistency. *Comput. Graph. Forum* 28, 3 (2009), 831–838.

[125] N. Smirnov. 1948. Table for estimating the goodness of fit of empirical distributions. *Ann. Math. Stat.* 19, 2 (1948), 279–281.

[126] Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.* 45 (2009), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

[127] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. 2018. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Trans. Vis. Comput. Graph.* 24, 1 (2018), 667–676. https://doi.org/10.1109/TVCG.2017.2744158

[128] G. K. L. Tam, V. Kothari, and M. Chen. 2017. An analysis of machine- and human-analytics in classification. *IEEE Trans. Vis. Comput. Graph.* 23, 1 (2017), 71–80. https://doi.org/10.1109/TVCG.2016.2598829

[129] Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proceedings of the Annual Conference of the Association for Computational Linguistics (ACL'02).* Association for Computational Linguistics, Stroudsburg, PA, 120–127. https://doi.org/10.3115/1073083.1073105

[130] Andrada Tatu, Georgia Albuquerque, Martin Eisemann, Peter Bak, Hogler Theisel, Marcus Magnor, and Daniel Keim. 2010. Automated analytical methods to support visual exploration of high-dimensional data. *IEEE Trans. Vis. Comput. Graph.* (2010), 1–14. https://doi.org/10.1109/TVCG.2010.242

[131] C. Tominski, G. Fuchs, and H. Schumann. 2008. Task-driven color coding. In *Proceedings of the 2008 12th International Conference Information Visualisation.* 373–380. https://doi.org/10.1109/IV.2008.24

[132] Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.* 2 (Mar. 2002), 45–66. https://doi.org/10.1162/153244302760185243

[133] Devis Tuia, Michele Volpi, Loris Copa, Mikhail Kanevski, and Jordi Munoz-Mari. 2011. A survey of active learning algorithms for supervised remote sensing image classification. *IEEE J. Select. Top. Sign. Process.* 5, 3 (2011), 606–617.

[134] S. van den Elzen and J. J. van Wijk. 2011. BaobabView: Interactive construction and analysis of decision trees. In *Proceedings of the IEEE Visual Analytics Science and Technology (VAST'11).* 151–160. https://doi.org/10.1109/VAST.2011.6102453

[135] Jeroen Vendrig, Ioannis Patras, Cees Snoek, Marcel Worring, Jurgen den Hartog, Stephan Raaijmakers, Jeroen van Rest, and David A. van Leeuwen. 2002. TREC feature extraction by active learning. In *Proceedings of the Text REtrieval Conference (TREC'02).*

[136] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin. 2017. Cost-effective active learning for deep image classification. *IEEE Trans. Circ. Syst. Vid. Technol.* 27, 12 (Dec. 2017), 2591–2600. https://doi.org/10.1109/TCSVT.2016.2589879

[137] Meng Wang and Xian-Sheng Hua. 2011. Active learning in multimedia annotation and retrieval: A survey. *ACM Trans. Intell. Syst. Technol.* 2, 2, Article 10 (2011), 10:1–10:21 pages. https://doi.org/10.1145/1899412.1899414

[138] Yunhai Wang, Zeyu Wang, Tingting Liu, Michael Correll, Zhanglin Cheng, Oliver Deussen, and Michael Sedlmair. 2020. Improving the robustness of scagnostics. *IEEE Trans. Vis. Comput. Graph* 26, 1 (2020).

[139] Joe H. Ward. 1963. Hierarchical grouping to optimize an objective function. *J. Am. Stat. Assoc.* 58, 301 (1963), 236–244. http://www.jstor.org/stable/2282967.

[140] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. 2019. The What-If Tool: Interactive probing of machine learning models. *IEEE Trans. Vis. Comput. Graph.* (2019), 1–1. https://doi.org/10.1109/TVCG.2019.2934619

[141] Leland Wilkinson, Anushka Anand, and Robert L. Grossman. 2005. Graph-theoretic scagnostics. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'15).* 157–164. https://doi.org/10.1109/INFOVIS.2005.14

[142] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mané, D. Fritz, D. Krishnan, F. B. Viégas, and M. Wattenberg. 2018. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE Trans. Vis. Comput.Graph.* 24, 1 (2018), 1–12. https://doi.org/10.1109/TVCG.2017.2744878

[143] Yi Wu, Igor Kozintsev, Jean-Yves Bouguet, and Carole Dulong. 2006. Sampling strategies for active learning in personal photo retrieval. In *Proceedings of the IEEE International Conference on Multimedia and Expo.* IEEE, 529–532. https://doi.org/10.1109/ICME.2006.262442

[144] Bo Yu, Mingqiu Song, and Leilei Wang. 2009. Local isolation coefficient-based outlier mining algorithm. In *Proceedings of the Conference on Information Technology and Computer Science (ITCS'09).* IEEE Computer Society, 448–451. https://doi.org/10.1109/ITCS.2009.230

[145] J. Zhang, X. Wu, and V. S. Sheng. 2013. A threshold method for Imbalanced Multiple Noisy Labeling. In *Proceedings of the IEEE ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'13).* 61–65. https://doi.org/10.1145/2492517.2492640

[146] J. Zhang, X. Wu, and V. S. Shengs. 2015. Active learning with imbalanced multiple noisy labeling. *IEEE Trans. Cybernet.* 45, 5 (May 2015), 1095–1107. https://doi.org/10.1109/TCYB.2014.2344674

[147] Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. 2019. Interpreting cnns via decision trees. In *Computer Vision and Pattern Recognition.* 6261–6270.

Received November 2019; revised September 2020; accepted November 2020