Visual Analytics for Classifier Construction and Evaluation for Medical Data

Jacek Kustra and Alexandru Telea

1 Introduction

In the last decade, machine learning (ML) has made tremendous progresses and inroads into a wide range of application areas, including image classification, time series prediction, and text pattern mining, with application to several fields such as social networks [43], automotive self-driving [30], and, last but not least, medical science [1].

An important problem that ML addresses is that of *classification*: Given a set of observations, the goal is to assign a label from a (typically small) predefined set to each observation, based on the similarity of such observations with those from a so-called training set. Classification is central to medical tasks such as diagnosis [29] and prognosis [1] of various types of diseases based on clinical patient data.

Classification methods can be roughly divided into two main types, as follows:

Deep learning techniques based on artificial neural networks (ANNs) are the more recent introductions to the field and have shown strong advantages for such classification tasks, as they require minimal user intervention and fine-tuning [48]. In many cases, one can simply feed the training and/or test data at hand to such a network and largely rely on the network's inherent flexibility for learning relevant features to perform the desired classification. Recent results show very high classification accuracy for complex problems and datasets [20]. However, ANNs also have fundamental limitations: They typically require a very high number of labeled observations for training, in the order of tens of thousands or even more.

J. Kustra (✉)
Philips Research, Eindhoven, The Netherlands
e-mail: jacek.kustra@philips.com

A. Telea
Institute Johann Bernoulli, University of Groningen, Groningen, The Netherlands
e-mail: a.c.telea@rug.nl

S. Consoli et al. (eds.), *Data Science for Healthcare*,
https://doi.org/10.1007/978-3-030-05249-2_10

Obtaining such labeled datasets can be impractical or even impossible in certain medical contexts, e.g., where observations are patients having a rare condition and/or when labeling incurs high manual effort [5]. In addition, the understanding of the model's intrinsic working and the assumptions underlying the relationships between features can be of key importance to ensure human (domain) knowledge and supervision are taken into account when constructing a model and also to convey trust in how the model operates.

Explicit features are the more traditional classifier engineering methods. Here, the classifier designer explicitly specifies how to extract several features (also called dimensions or variables) from the input data, following established insights and practices in a given field on which aspects of the data are discriminative for the different classes of interest. Using classifiers based on *explicit features* can be more effective than using ANNs. However, this approach has its own challenges: Simple rule-based models (a subclass of explicit-feature classifiers) are usually defined based on vague heuristics; and mixing domain expert knowledge with data insights is a complex task as it requires "showing" the domain expert how the data is actually used by the model. Applying all above in practice is hard as several questions need to be answered, regarding what is the nature of hard-to-classify observations, which classification technique is the best and why, and how to set its parameters. Exploring the high-dimensional space spanned by all these choices, a process we next call *classifier engineering*, is very challenging, time consuming, and error prone [26].

Visual analytics (VA) addresses the problem of understanding large amounts of high-dimensional unstructured data by interactive and iterative exploration of depictions of such data [24, 25]. As such, VA can be an important instrument in the toolset of engineering classifiers based on explicit features. Recent efforts indicate promising results for combining ML and VA techniques for classifier engineering [45].However, to date, VA has been rarely documented in how it supports this process *end-to-end*, i.e., covering all the steps of dataset structure exploration, feature assessment and selection, classifier accuracy comparison, and classifier improvement. One key reason for this is that ML and VA have evolved historically separately, with limited cross-discipline dissemination.

In this work, we extend the recent VA approach and VA toolset of Rauber et al. for explicit-feature classifier engineering [45] in two main directions:

- We extend the functionality of the abovementioned toolset with additional classifiers, feature selection methods, and manual data clustering methods;
- We present a detailed step-by-step application of this toolset to the problem of engineering a classifier for predicting biochemical recurrence, an indicator of potential cancer relapse after prostate cancer treatment, from clinical patient data. This presents concrete evidence of the added value of our approach and also provides a practical example of how to cover all the steps required for effectively and efficiently using VA in such a classifier engineering problem in a real-world medical context.

2 Related Work

We outline related work in ML and VA along two main axes: classifier design and visual analytics for classifier design, as follows.

Classifier Design Let $D = \{\mathbf{d}_i\}, 1 \leq i \leq n$ be a set of observations, or samples $\mathbf{d}_i = (d_i^1, \ldots, d_i^m)$ taken from a m-dimensional space $\mathscr{D}$, where d_i^j are the so-called dimensions, or features, of a sample. We denote by the feature vector $\mathbf{f}^j = (d_1^j, \ldots d_n^j)$ the values of feature j over all samples and by $F = \{\mathbf{f}^j\}$, $1 \leq j \leq m$, the set of all m feature vectors. Feature values d_i^j can be either quantitative (real) values or categorical values. Let L be a set of categorical labels or classes. Briefly put, the problem of designing a classifier for $\mathscr{D}$ is to find a function $f : \mathscr{D} \rightarrow L$ which associates to any sample in $\mathscr{D}$ a label in L. To design f, one typically uses a training set of labeled samples $D_t = \{(\mathbf{d}_i, l_i)\} \subset \mathscr{D} \times L$, $1 \leq i \leq n$, to maximize the number of samples in D_t for which $f(\mathbf{d}_i) = l_i$. Different optimization methods give birth to different classification techniques, such as k nearest neighbors (KNN) [3], random forest classifiers (RFC) [12], support vector machines (SVM) [8], and learning vector quantization (LVQ) [27]. To test f, one typically counts, for a test set of labeled samples $D_T | D_T \cap D_t = \varnothing$, the number of correctly labeled samples $\mathbf{d}_i \in D_T | f(\mathbf{d}_i = l_i)$. Besides this simple so-called classifier *accuracy*, more complex measures can be used, such as the area under the receiver operator curve (AUROC) [15].

The challenges of developing a good classifier—finding a f which yields high accuracy and/or AUROC values—can be grouped into intrinsic and technical ones. *Intrinsic* challenges relate to the availability of a "good" set of features $\mathbf{f}^j$ which capture differences between the different classes, the availability of a sufficient number of diverse samples that cover well the underlying phenomenon that we wish to classify, and the accuracy of feature measurements f_i^j and assigned labels in D_t. We call these challenges intrinsic since one cannot typically alleviate such issues by changing the classifier technique and/or its parameters. *Technical* challenges relate to the choice of optimization method and optimization parameters used to compute f—or, in more familiar words, how one preprocesses and/or selects the features, samples the hyperparameter space of f, and chooses the actual classification technique f. Intrinsic challenges are often outside the full control of the classifier engineer. In contrast, the technical challenges can be seen as a meta-optimization problem: How can we support the engineer in the process of design, training, and testing a classifier, so as to obtain maximal accuracy results with minimal effort?

Visual Analytics for Classifier Design Aware of the abovementioned challenge of classifier design, also called the "black art" of, or opening the "black box" of, classifier design [11, 36, 38, 53, 57], several types of methods have been proposed to help various steps of classifier engineering. The most common techniques include correlation analysis, displayed, e.g., by matrix plots, to show the correlation of any pair of features $(\mathbf{f}^i, \mathbf{f}^j)$; and ROC graphs to show how specificity and sensitivity are related. Dimensionality reduction (DR) techniques, also called projections, such

as PCA [22], LAMP [21], or, more recently, t-SNE [55], are used to show the so-called structure of the input data D by means of 2D scatterplots where inter-point distance reflects sample similarity in $\mathscr{D}$, helping one to correlate sample clusters with their assigned labels and thus detect the kind(s) of observations that are hard to classify [4, 31, 32, 49, 56]. Given the recent popularity of ANNs, specialized visual analytics techniques have been designed for these architectures, to explore, e.g., the activation patterns of hidden-layer neurons [46] or to find problems in the network design during training [44]. A recent survey of VA techniques for deep learning is given in [19]. While being good examples of the added value of VA for machine learning, such techniques are not applicable to more classical designs, such as KNN, RFC, SVM, or LVQ, which we consider in our work.

For such architectures, *features* play a key role in the analysis, as one aims to understand how they correlate with each other but also how their values affect the similarity of and, ultimately, the labels assigned to samples. For these ends, specific techniques have been designed. Confusion matrices are used to compare the performance of different classifiers [52]. DR methods can be modified to implicitly label unsupervised clusters with the identities of their most discriminative features [9]. More involved toolsets aim to cover several of the classifier engineering steps. Early on, RadViz [17] proposed a DR technique where one can see both the data structure (clusters) and how all features affect their appearance. Atop of this, clustering techniques are provided to explicitly segment D into sets of similar observations; feature scoring, based on the t statistic, which ranks how important a feature $\mathbf{f}^j$ is to samples having a given class l_i as opposed to samples of all other classes $l_{k\neq i}$, allows users to eliminate features which do not strongly help classification. However, RadViz has several limitations: (1) its DR method preserves sample similarity far less than state-of-the-art techniques such as LAMP or t-SNE; (2) feature scoring is used only to order features, yielding different scatterplots of the input data; mechanisms for actual feature selection are not provided; (3) visual data exploration is not integrated with actual classifier construction, training, and testing, which breaks end-to-end support for classifier engineering (Sect. 1). RadViz's limitation (1) above was alleviated by the VizRank [28] and FreeViz [10] tools which added the ability to select DR scatterplots which best visually discriminate between classes. However, limitations (2) and especially (3) are still present in these tools.

The above limitations of RadViz and its followers are alleviated by a recent toolset for classifier engineering proposed by Rauber et al. [45]. The least square projection (LSP) method [41] is used for constructing DR scatterplots, which gives a better data structure preservation than the earlier techniques used in [10, 17, 28]. Instead of RadViz's simple t test, more advanced feature scoring techniques including univariate ones (χ^2, one-way ANOVA), multivariate ones (IRelief [51]), and classifier wrappers (ensembles of randomized decision trees [12], randomized logistic regression [34], and recursive feature elimination [14]) are used. These allow users to interactively select features which characterize well-specific sample clusters. As demonstrated in [45], this toolset effectively supports reducing the dimensionality of an input dataset (by feature elimination) before training a classifier on it.

3 Part 1: Visual Analytics Toolset and Workflow

We next describe the original toolset of Rauber et al. [45] and our implemented extensions (Sect. 3.1) and outline the workflows supporting classifier engineering that our extended toolset, called *featured*, supports (Sect. 3.2).

3.1 Featured *Toolset*

Original Tool The tool in [45] provides several interactive views for data exploration and analysis—see all views in Fig. 1 except the Feature view, which we added in this work. These work as follows. The tool reads as input a sample dataset D stored in simple CSV matrix format (samples $\mathbf{d}_i$ are rows, features $\mathbf{f}^j$ are columns). Upon loading D, the observations $\mathbf{d}_i$ are displayed in the Observation view as text items, or, if image tags are provided for these, as thumbnails, and the names of the features $\mathbf{f}^j$ are listed in the Feature selector view. Both these views allow selecting a subset of samples $S_D \subset D$ or of features $S_F \subset F$ to work with next. The Observation map displays all selected samples S_D as a 2D scatterplot, using PCA or LSP as projection technique. Samples can be colored by the value of a selected feature $\mathbf{f}^j$, or class label. This allows seeing whether there is apparent structure in D, e.g., in terms of clusters or outliers. To explain which features determine such structure, one can next select S_D in the Observation view (see the dark red points in Fig. 1) and invoke the Feature scoring view, which displays, for all features $\mathbf{f}^j \in F$, a score indicating how much each $\mathbf{f}^j$ contributes to the separation between S and $D \setminus S$. Scores are computed by various scoring techniques, as explained in Sect. 2. Features are shown in the Feature scoring view as bars scaled and sorted by score

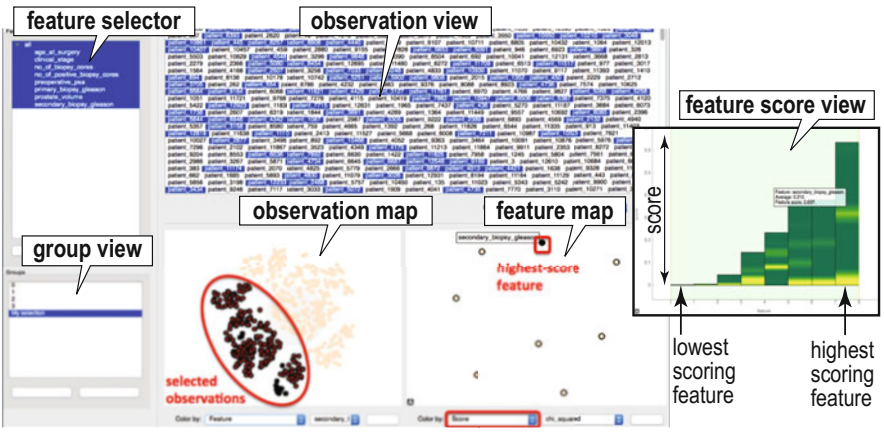

Fig. 1 *featured* toolset for classifier engineering using visual analytics (Sect. 3.1)

and colored by the frequency of samples over the entire range of a given feature using a green (low) to yellow (high) colormap. For instance, in Fig. 1 we see that the highest scoring feature (rightmost bar) has mostly low and mid-range values (yellow at the bottom and halfway that bar, green for the rest of the bar). The Feature scoring view also allows selecting a subset of features S_F to work with next, upon which the Observation view updates to project D only considering these features. Finally, the Group view allows saving selected sample subsets S_D under given names, for further analysis.

Tool Extensions Overall, the original tool [45] allows a flexible way to *explore* the structure of a high-dimensional dataset D in terms of finding sample clusters or outlier samples and *explain* these by means of relevant features and/or feature values. While useful, however, such actions do not fully support the end-to-end classifier engineering pipeline. To this end, we extended the tool in the following three main directions:

- *Classifiers:* We integrated five types of classifier techniques in the tool: KNN, RFC, SVM with linear and radial basis functions (SVM-L, SVM-R), logistic regression (LR), and two LVQ variants. To use any of these, the user can interactively select the training and test sets in the tool's various views, run k-fold cross-validation, examine the misclassifications in the Observation view, and examine the overall accuracy and AUROC metrics. For small datasets (up to 20,000 observations, 10–20 dimensions), the current implementation performs such operations in under 10 s on a modern PC. All classifiers accept data which can be normalized either by scaling or standardization (see next Sect. 4.5) and can use various similarity metrics—Euclidean, cosine, or learned distances (LVQ).
- *Projections:* We extended the original tool by adding IDMAP [35], Sammon mapping [47], LAMP [21], and t-SNE [55] as projection techniques. This is important, since, as known in projection literature, no single projection technique performs well (in terms of preserving the data structure) on any type of dataset [4, 49, 56]. In particular, t-SNE has shown to be a very effective predictor of the ease of classifying data [54].
- *Feature map:* To better understand how different features correlate with each other and contribute to the data structure, we provide a new Feature map view (see Fig. 1. Every point in here is a feature vector $\mathbf{f}^j \in F$. The points are placed based on a 2D projection of the set F, using as similarity metric the Pearson correlation or Spearman's rank between these feature vectors. Hence, close points in this plot indicate strongly similar features over the entire sample set D, while far away points indicate independent features. Separately, points are colored to depict the scoring of all features for the discrimination between a selected sample set S_D and the remaining samples $D \setminus S_D$. In other words, this view enhances the Feature scoring view by showing not only which features discriminate most between S_D and $D \setminus S_D$ but also how these features are correlated. We show next how this information is helpful in classifier engineering.

3.2 Visual Analytics Workflow

Explaining a VA workflow is, in general, hard [24, 25]. Yet, in our classifier engineering context, the key elements of our VA approach are as follows:

- Show the data at hand (D) and its classes L and how, where, and why these do or do not correlate. This way, engineers can see whether and how D is "partitioned" into different groups (clusters) of similar observations, and whether there is a correlation between these clusters and their labels; lack of such (strong) correlations indicates for which observations and/or which labels we will expect classification problems;
- Show which features $\mathbf{f}^j$ of our dataset D are most responsible for correlations of observations with label values. This helps understanding the predictive power of different features;
- Show how feature engineering effectively influences classification accuracy. This way, one can navigate the design space of the classifier, understanding easier which feature-engineering actions were useful (in increasing accuracy, and for which observation or label types) and which not.

The way in which VA supports all the above tasks, and is therefore instrumental in helping classifier engineering, is illustrated next via a concrete, real-world application.

4 Part 2: Application in Predicting Biochemical Recurrence After Prostate Cancer Treatment

4.1 Motivation

Predicting the evolution of medical conditions in terms of different metrics such as relapse, survival, or quality of life following a given treatment can provide vital information to select the optimal treatment for a particular patient. Having this prediction available for several treatment options can provide insights into which treatment is optimal for the specific patient. In particular, for a given treatment, being able to infer the progression of a certain disease based on the patient's clinical and disease-specific diagnostic information can save large amounts of effort, cost, and patient well-being especially in the early stages of the disease's evolution. Such is the case for prostate cancer. After patients diagnosed with this cancer type are treated, a treatment (or lack of it, by assigning it with active surveillance) plan is defined for the patient taking into account the available medical information and patient preferences. Treatment options typically involve surgery (prostatectomy), chemotherapy, radiation therapy, or a combination therapy involving two or more of the above options. Following treatment, the increase in concentration of a prostate-specific antigen (PSA), a phenomenon called biochemical recurrence (BCR), is a

good indicator for potential cancer recurrence, either in the prostate or other parts of the body. Since BCR typically appears earlier than other signals that diagnose cancer relapse by several years, predicting its appearance can save precious time for controlling, or preventing, the evolution of the disease [39, 50]. Therefore, the measurement of BCR typically happens at discrete points in time following treatment. Since BCR is a time-dependent outcome, for the purpose of this study, we define two classes: 0—no recorded relapse after treatment, or 1—relapse recorded after 5 years following treatment.

Given the influence a prediction of BCR can have on the medical decision for a patient based on the information present prior to treatment, several research questions emerge:

- Is it possible to reliably predict BCR values from the above measurements?
- Which of the above measurements are the most discriminative in predicting specific BCR values?

If answered positively, the first question indicates that "standardized" decision-support systems can be offered to physicians so that they profit from the knowledge captured by such systems which, in general, can be wider and/or more diverse than their personal experience. Separately, if we have ways to objectively and intuitively answer the second question, this will increase the confidence (and ultimately the adoption rate) of such automated decision-support systems by medical specialists. All in all, this has the potential to increase the efficiency and/or effectiveness of diagnosis and treatment of prostate cancer, with important cost savings and/or quality improvement as outcomes.

In this section, we detail the engineering of a set of classifier systems for predicting BCR values from clinical measurements for prostate cancer. Key to this is our use, during the whole process, of the visual analytics (VA) techniques provided by the *featured* toolset introduced in Sect. 3 for data exploration and classifier construction, testing, and improvement. We next describe these steps, as well as our obtained results. For each step, we outline the relevant questions to be solved and how VA assisted in answering these to lead to the next step.

4.2 Data

The input data (used next for training and testing the classifier) consists of a set D of prostate cancer patients where for each patient, a total of $m_{\text{total}} = 50$ features are measured. The actual clinical measurements took place over different periods in time and were performed by an unknown number of different medical specialists. From these $m = 50$ values, we next manually selected a small subset of $m = 9$ features (see Table 1) to use next in predicting the presence of biochemical recurrence (BCR) within a period of 5 years from the measurement moment. The selection was based on the type of features which are, to our knowledge, widest available and easiest to measure in medical practice. Hence, ground truth

Table 1 Input data for prostate cancer prediction (Sect. 4.2)

Feature name	Feature type	Feature range
Age at surgery	Quantitative	[37.6,78]
Prostate volume	Quantitative	[9,365]
Preoperative PSA level	Quantitative	[0.11,107.11]
Number of biopsy cores	Integral	[1 ... 28]
Number of positive biopsy cores	Integral	[1 ... 10]
Positive biopsy cores (%)	Quantitative	[10,90]
Primary biopsy Gleason score	Integral	[2 ... 5]
Secondary biopsy Gleason score	Integral	[2 ... 5]
Clinical stage	Ordinal	{T1, T1a, T1b, T1c, T2, T2, T2b, T2c T3, T3a, T3b, T3c}

is available for the data in terms of two class labels—patients showing, respectively not showing, BCR within 5 years from measuring the nine features. Given this data, we want to construct a classifier able to accurately predict these two classes.

4.3 Preprocessing

To make the data directly usable, we first eliminate all samples (rows in D) where at least one of the nine columns of interest (eight features plus class label) misses the values. The second step regards the treatment of the *clinical stage* feature. As shown in Table 1, this is an ordinal variable taking values over the three stages T1, T2, and T3; the sub-labels (a, b, c) indicate gradations within each major stage; values having no sub-label, e.g., T1, indicate that for that patient no finer-grained information is available. We convert these ordinal values into quantitative ones by using

$$Tij = \alpha(i-1) + \beta val(j), \tag{1}$$

where $val(a) = 1$, $val(b) = 2$, $val(c) = 3$, and $val(empty) = 0$, where $empty$ designates entries for which we have no sub-label value, e.g., T1. The parameters $\alpha > 0$ and $\beta > 0$ with $\alpha > \beta$ control the relation between the importances of the major stages (T1, T2, T3) to that of the importances of the sub-stages (a, b, c). We set by default $\alpha = 10$ and $\beta = 1$. The effect of these two parameters is discussed in detail next in Sect. 4.6. With this conversion, we have now a fully quantitative dataset which we can use for classifier engineering, as described next.

4.4 First Exploration: How Hard Is the Classification Problem?

Before actually aiming to build (train) a classifier, we want to assess how hard the classification problem may be and how the available eight features contribute to the separation of the two classes. For this, we project all the available samples using t-SNE, as it is known that this method achieves a quite good separation of existing data clusters [55], and color the projected samples by their two class labels (Fig. 2a). We see that there is no clear separation between the blue (no BCR within 5 years) and orange (BCR within 5 years) samples. This already indicates a hard classification problem ahead of us. Next, we select all points of one class and construct the feature map using as feature similarity the Pearson correlation and as feature scoring technique the χ^2 test, respectively (Sect. 3.1). The resulting image (Fig. 2b) shows us three insights: (1) We see that there are no strongly correlated features, except the total number and percentage of positive biopsy cores, whose respective points are relatively close in the map. This indicates that, within our eight feature set, there are no obviously redundant features. (2) The number of samples is quite unbalanced—there are many more blue than orange ones. This will need to be considered when engineering the classifier. (3) We next see that only a subset of features have high scores (dark red points in the map). This suggests that we could drop the other features (brighter-color points) from our dataset without reducing the chances of building an accurate classifier. However, we need to further check this hypothesis. For this, we use the feature scoring view, with ensembles of randomized

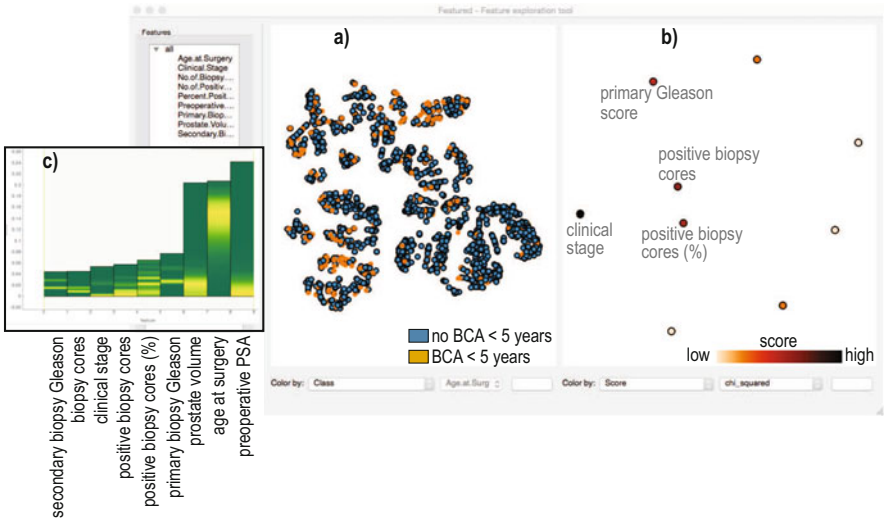

Fig. 2 First visual exploration of the input data (Sect. 4.4). (**a**) Observation view, (**b**) feature map, (**c**) feature score view

decision trees [12] as scoring technique (Fig. 2c). As visible, the relative scores of the most discriminating features are now very different as compared to the χ^2 scoring technique used earlier. This indicates that we cannot, so far, drop any of the available eight features for being not useful for classification. Separately, this indicates that the type of considered scoring function, thus implicitly the *distance metric* used to compare samples, is very important. We will revisit this insight later on.

4.5 Classifier Design: First Experiments

Based on the insights learned during the first visual exploration (Sect. 4.4), we next proceed to the actual training and testing a classifier, as follows. We first extract a balanced dataset from the input data, based on insight (2) found earlier, using random sample selection from the larger class. With this dataset, we next train and test four different classifiers (KNN, RFC, SVM-R, SVM-L), and we also consider a dummy classifier, for sanity checking. Optimal classifier parameters are found by grid search using the classifier accuracy acc (number of correctly classified samples divided by total sample count) as optimization criterion. For testing, we use fivefold stratified cross-validation with a split of 66% to 33% between training and test data. For normalization of the different features (columns), we use both scaling and standardization.

Table 2 shows the obtained accuracy results from this first experiment. As visible, the standardization normalization is slightly but consistently better than the scaling normalization. As such, we use this next as default in our designs. As expected, the dummy classifier returns an accuracy of 50%, which tells us that our testing pipeline is correctly set up. Most importantly, we see that the classification accuracy is quite independent on the classifier method and also relatively low. Hence, we ask ourselves next which steps can be taken to improve this accuracy.

Table 2 Classifier accuracy for first design (Sect. 4.5)

Standardization normalization		Scaling normalization	
Classifier technique	Accuracy	Classifier technique	Accuracy
KNN	69.853	KNN	69.345
RFC	66.878	RFC	66.369
SVM-R	66.666	SVM-R	66.634
SVM-L	65.423	SVM-L	65.201
Dummy	50.000	Dummy	50.000

4.6 *Classifier Refinement: What Can We Do Better?*

To improve our accuracy results, several directions can be considered. A first and quite obvious one relates to our initial decision of converting the categorical clinical stage values into quantitative ones (Eq. (1)). Before actually trying to find better values for the α and β parameters, let us see how the engineered quantitative *clinical stage* feature given by Eq. (1) correlates with the class labels and classification results. For this, we use the observation view to project our balanced dataset using again t-SNE, and color the samples by classification correctness (Fig. 3a), next by the ground-truth labels (Fig. 3b), and finally by the values of the *clinical stage* feature computed with the defaults $\alpha = 10$ and $\beta = 1$ (Fig. 3c). We find several insights by studying these plots. First, we see that the data appears to be separated in three large clusters Γ_1–Γ_3, each consisting of two smaller sub-clusters (see outlines in Fig. 3a). However, these clusters do not correlate in any way with the class labels (Fig. 3b). Moreover, the classification errors are equally spread over these clusters (Fig. 3a). Yet, the clusters correlate quite well with the value of the *clinical stage* feature—high values in the two top clusters Γ_1 and Γ_2, low values in the bottom one Γ_3 (Fig. 3c). This suggests that the engineered feature may influence the data structure in a too strong, and actually undesired, way that does not help the classification.

To further understand this, we test and train our classifiers using different values for α and β in Eq. (1). As we aim to visually explore these results at near-interactive rates, we do not perform now the more costly fivefold cross-validation used earlier (Sect. 4.5), but run a single test-train experiment, which takes only a few seconds. Figure 4 shows the observation views for five (α, β) combinations, for the RFC classifier, ranging between very strong differences considered between the major clinical stages T1, T2, and T3 ($\alpha = 100, \beta = 1$), through moderate differences ($\alpha \in \{3, 10\}, \beta = 1$), no differentiation between sub-stages ($\alpha = 1, \beta = 0$), and completely dropping this feature ($\alpha = 0, \beta = 0$). Similar results to Fig. 4

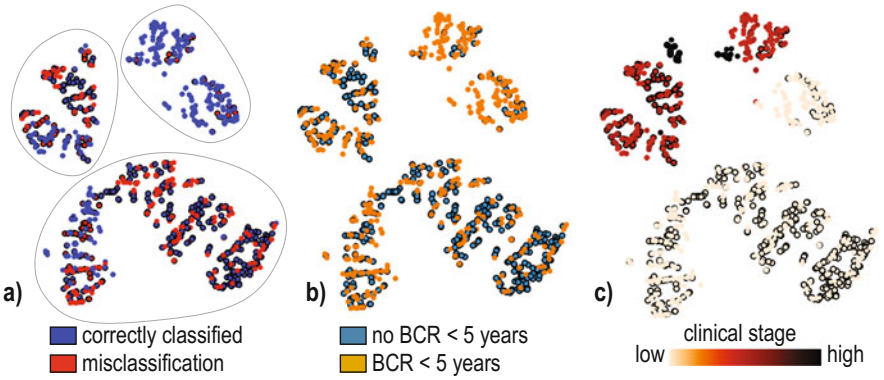

Fig. 3 Understanding the distribution of the engineered *clinical stage* feature (Sect. 4.6)

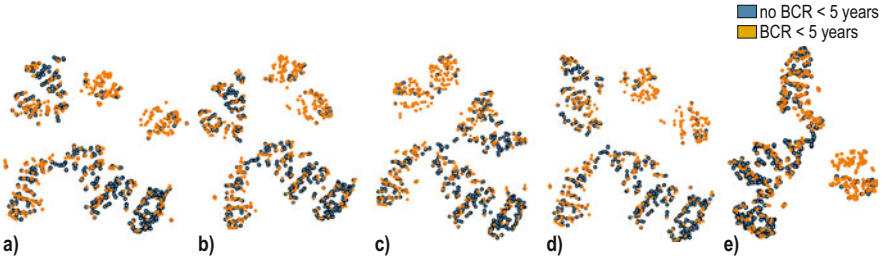

Fig. 4 Understanding the parameters α and β of the engineered *clinical stage* feature (Sect. 4.6). (**a**) $\alpha = 100, \beta = 1$; Tij $\in$ [0, 1, 2, 3, 100, 110, 120, 130, 200, 210, 220, 230]; $acc = \mathbf{63.048}\%$, (**b**) $\alpha = 10, \beta = 1$; Tij $\in$ [0, 1, 2, 3, 10, 11, 12, 13, 20, 21, 22, 23]; $acc = \mathbf{63.048}\%$, (**c**) $\alpha = 3, \beta = 1$; Tij $\in$ [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]; $acc = \mathbf{63.147}\%$, (**d**) $\alpha = 1, \beta = 0$; Tij $\in$ [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2]; $acc = \mathbf{62.351}\%$, (**e**) $\alpha = 0, \beta = 0$; Tij $\in$ [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]; $acc = \mathbf{62.849}\%$

are obtained for the other considered classifiers (omitted here for brevity). These images give us additional insights, as follows. First, we see that the obtained accuracy values are lower—roughly 63 vs 66–69%—than those obtained when using the more exhaustive evaluation discussed in Sect. 4.5. This is expected, given the rapid training-testing procedure explained above. More interestingly, we see that the α and β settings appear to not significantly affect the class separation nor the classification accuracy. This suggests that the *clinical stage* feature is completely non-discriminative for the two considered classes. However, we have seen that this feature scores quite high discrimination-wise (χ^2 test, Fig. 2b). Putting these two insights together, we formulate the hypothesis that the problem (of relative insensitivity of the RFC classifier to the *clinical stage* feature) is due not so much to the engineering of this feature (α and β values), but to the distance metric that this feature is next used with inside the classifier.

To test this hypothesis, we next examine how the *range* of the Tij values is correlated to the classification accuracy. As we have seen in Fig. 3, the samples can be split into three groups Γ_1–Γ_3, where only Γ_1 has high T-value samples—more precisely, Tij equal to values in the T2 and T3 stages. Let us now select all samples in Γ_1 having such high T-values (Fig. 5b) and remove these from the dataset, by interactively selecting the dark-colored points in the observation view in *featured*. The remaining points are shown in Fig. 5c. We now run the same classification procedure on this subset of points and obtain a larger accuracy ($acc = 65.379\%$ vs $acc = 63.546\%$. Interestingly, the misclassifications are not correlated with the T-value distribution in *neither* the initial dataset nor the dataset with removals—see the uniform spread of blue and red points in both Fig. 5a, c. We have now a number of interesting findings: (1) The analysis in Sect. 4.4 showed us that *clinical stage* can be highly discriminative between our two classes, depending on the considered distance function. (2) The current analysis showed us that samples with high T-values confuse the classifier.

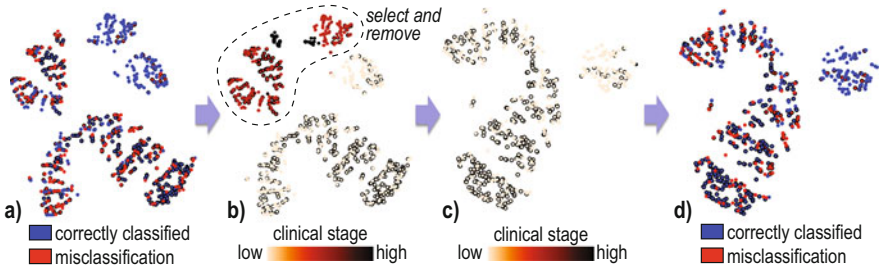

Fig. 5 Understanding how different ranges of the engineered *clinical stage* feature affect classification accuracy for the RFC classifier (Sect. 4.6). (**a**) All data: $acc = 63.546\%$, (**b**) find high T-value samples, (**c**) remove these samples, (**d**) remaining data: $acc = 65.379\%$

Taken together, we formulate the hypothesis that one issue with the current set-up is a suboptimal *distance function* used internally by the considered classifiers. So far, we have used the Euclidean m-dimensional distance metric (on the standardized data values), which is the default in *featured*. We next run the same classification experiment as in Fig. 5a, but using the cosine distance metric, and use all available classifiers in our tool. We obtain the following accuracy values: 66.932% (KNN), 68.147% (RFC), 68.526% (SVM-R), and 68.825% (SVM-L). These are all (slightly) higher than the accuracy obtained by using the Euclidean metric (63.546%, RFC). Hence, we validate the hypothesis that the distance metric used has a *clear* effect on classification accuracy.

This finding leads us to the final refinement in our classifier design: We consider using Generalized Matrix Learning Vector Quantization (GMLVQ) [16], a variant of the classical LVQ classifier [27] which is able to learn the distance function from the training set. GMLVQ works as follows (for full details, we refer to [16]): We firstly define a set of so-called prototypes $\mathbf{w}_i \in \mathbb{R}^m$. Secondly, we associate a (typically equal) number of prototypes with each class. Thirdly, during training, prototypes are moved in $\mathbb{R}^m$ so that their nearest neighbors from the training set match their class labels, using a gradient-descent optimization process. Atop this process offered by LVQ, GMLVQ also allows learning the distance metric $d(\mathbf{x}_j, \mathbf{w}_i)$ used to compare a training sample $\mathbf{x}_j$ with a prototype $\mathbf{w}_i$, defined as

$$d(\mathbf{x}_j, \mathbf{w}_i) = (\mathbf{x}_j - \mathbf{w}_i)^T A(\mathbf{x}_j - \mathbf{w}_i), \tag{2}$$

where A is a m-by$m$ real-valued distance matrix whose entries are learned during the aforementioned optimization process. If A is a diagonal matrix (as in classical LVQ), we obtain the classical Euclidean distance metric. Other values for A model distances where different features have different weights. Intuitively put, GMLVQ resembles a KNN classifier where the prototypes are the centers of several m-dimensional Voronoi cells, and all samples within a cell get the label of the cell's prototype. Given that A is not an identity matrix in GMLVQ, the boundaries of these cells can take complex shapes and therefore are able to approximate decision

boundaries better than the linear boundaries of LVQ. GMLVQ was shown in the past to yield good results for problems (datasets) where other classifiers did not perform well [16].

To assess the effectiveness of GMLVQ, we use again our balanced dataset that we considered so far. We train GMLVQ using two prototypes, one for each class. After training, we use the same dataset for testing, to assess the training errors. Moreover, we now perform a more detailed analysis of the quality of the classification, considering not only the aggregated accuracy but the finer-grained receiver operator curve (ROC). Figure 6 shows the obtained results. The first three images (a–c) show the evolution of the total training error, training error for the two classes, and area under the ROC (AUROC) as a function of the gradient-descent optimization iterations performed by GMLVQ, for 50 iterations. To construct the ROC, during the test phase, we consider that, for a GMLVQ classifier using two prototypes ($\mathbf{w}_1$ for class 1 and $\mathbf{w}_2$ for class 2), a test sample $\mathbf{x}$ is assigned to class 1 if

$$d(\mathbf{x}, \mathbf{w}_1) \leq d(\mathbf{x}, \mathbf{w}_2) - \theta, \tag{3}$$

and else to class 2. Here, θ represents the bias given to class 1, and d is given by Eq. (2). The fourth image (d) shows the final ROC obtained. We see how all error metrics converge quickly after roughly 30 iterations. We obtain an average error rate of 35% for the BCR within 5-year class and 25% for the no BCR within 5-year class, respectively (Fig. 6), yielding an aggregate average error of 30% for both classes (Fig. 6a). The corresponding AUROC value reached by optimization is 0.7624 (Fig. 6c). We evaluate the accuracy acc by selecting the point on the ROC corresponding to a bias $\theta = 0$ (Eq. (3)), i.e., for which GMLVQ assigns to a sample the label of the closest prototype (Fig. 6d, point marked $\theta = 0$). We obtain $acc = 75.2\%$. This is 10% higher than what we could obtain with all the earlier classifiers which used the Euclidean or cosine distances.

As these findings are encouraging, we aim to strengthen them by a deeper analysis. For this, we use again the balanced dataset, but perform now tenfolds of training and testing, with a 66% vs 33% training vs testing data split. Figure 7 shows the results. As visible, these are very similar to the training error analysis: GMLVQ

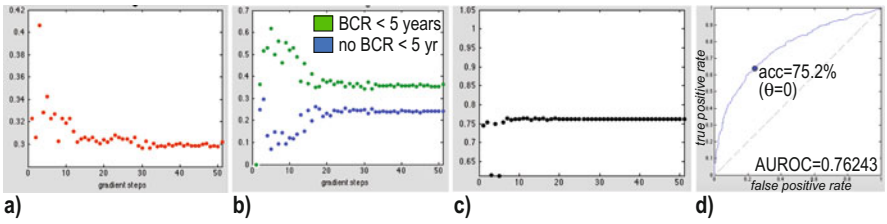

Fig. 6 GMLVQ training errors for balanced dataset. (**a**) Total training error, (**b**) per-class training error, (**c**) area under ROC (AUROC), (**d**) final ROC

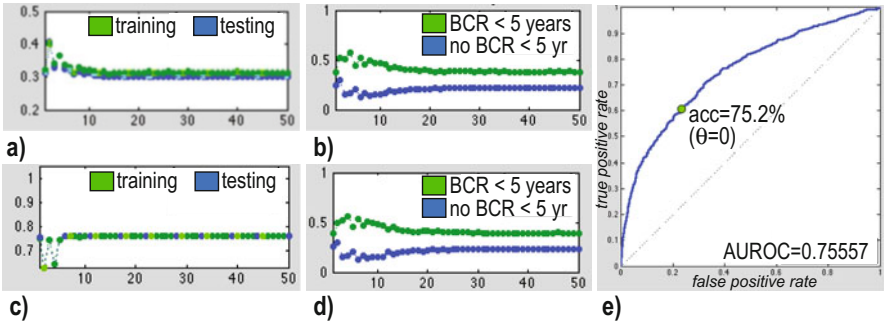

Fig. 7 GMLVQ training and testing errors for balanced dataset, tenfold cross-validation. (**a**) Total training and test errors, (**b**) per-class training errors, (**c**) AUROC, training and test sets, (**d**) per-class training errors, (**e**) final ROC (average, all folds)

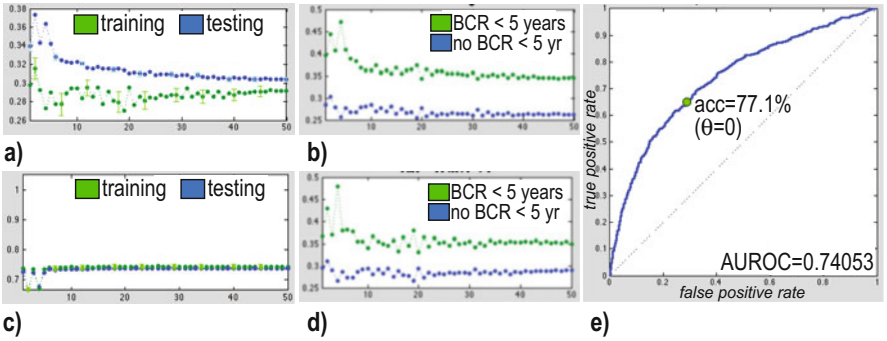

Fig. 8 GMLVQ training and testing errors for unbalanced (full) dataset, tenfold cross-validation. (**a**) Total training and test errors, (**b**) per-class training errors, (**c**) AUROC, training and test sets, (**d**) per-class training errors, (**e**) final ROC (average, all folds)

converges again quite quickly (25 iterations) and delivers an average error of 30% for both the training and test set. As before, the per-class errors (training and testing) are higher for the BCR within 5-year class (roughly 35% vs 25%, respectively). The AUROC values for training and testing are both 75.5%. Choosing again the point on the ROC in Fig. 7 for $\theta = 0$ (Eq. (3)), we obtain a classification accuracy of 75.2%.

To further confirm these good results, we finally consider the entire unbalanced dataset (see Sect. 4.4). We perform again tenfolds of training and testing, this time with a 33% vs 66% training vs testing data split. The training set is always balanced, randomly picked from the full dataset. In contrast to the previous experiments, we now use four prototypes for each of the two classes, in order to assess whether the performance of GMLVQ is affected by this choice. Figure 8 shows the results. Comparing these with Fig. 7, we find a slightly slower convergence requiring about 40 of the 50 iterations used. The average error (over both classes) is the same, roughly 30%, with a slightly different balance between the BCR within 5-year class (35%) and the no BCR within 5-year class (5%). This is explained by the way in

which the dataset is unbalanced. The average AUROC, however, is still quite good (0.74). For the chosen point on the AUC (Fig. 8e, $\theta = 0$), we obtain an accuracy of 77.1%, which is quite consistent (actually, slightly higher) than the value of 75.2% obtained for the previously considered balanced dataset.

In conclusion, the GMLVQ delivers the best results (accuracy of just over 77%) from all studied methods.

5 Discussion

We discuss next several relevant points related to our proposal of using visual analytics (VA) for classifier engineering.

Added Value of VA A very important question to answer is: What has been precisely the main added value of using VA in the process of classifier engineering for our application? The answer to this question is twofold. Firstly, VA provides to classifier designers *insights* on the consequences of all considered design choices (feature engineering, feature selection, and classifier design, training, and testing). This allows forming and testing hypotheses as to the optimality of a certain decision. When such decisions test positively, the respective design choices can be frozen and the design process advances to the next step. In the opposite case, the designer literally *sees* which are the undesired consequences of a design decision and can formulate hypotheses (new design choices) to next test. This way, VA "drives" the design process in a simpler and more controlled way than if one had to blindly choose directions for exploring the design space. Secondly, VA provides a way for actual end users of a classification system to visually understand how the system arrived at a given decision (label assignment) for a given observation. This can help the acceptance of such a system in decision-support contexts, especially when the end users are not machine learning experts.

Practically, using VA during our classifier engineering, we have been able to solve the problems of converting the clinical stage values and choosing the distance metric (and implicitly, classifiers that can handle this). Practically, all the experimental work described in this chapter has spanned under 10 h. This is far less than typically needed for refining classifier pipelines for similar contexts [13].

Related Workflows End-to-end workflow construction tools are becoming more and more pervasive in ML. For instance, RapidMiner [18] and KNIME [6] aim at roughly the same high-level end goal as our tool—to support the end-to-end data inspection, preprocessing, classifier engineering, validation, and refinement for a given problem domain. However, several differences exist between our tool and these. First and foremost, our VA approach, where the user is tightly integrated in an interactive sensemaking loop (observe the data, find patterns, change parameters of the pipeline, repeat until obtaining the desired result), is less present in these two tools, which advocate a more classical "waterfall" design. Second, our visualization options heavily rely on the use of multidimensional projections, and in particular

t-SNE, which have been found to be very well suited to explore high-dimensional data, especially when one wants to reason about observation groups. RapidMiner and KNIME, to our knowledge, do not offer t-SNE or such more advanced projections (with the exception of Self-Organizing MAps). Finally, they also do not incorporate some more advanced classification techniques, such as Generalized Matrix Learning Vector Quantization (GMLVQ). More importantly, as already explained, our main goal in this chapter is not to claim the superiority of any particular type of feature engineering, feature selection, or classifier technique, but to show how visual analytics can be the key element that efficiently binds all engineering actions together when designing a non-trivial classification system.

Limitations While useful, our VA proposal and its support in the *featured* toolset has several limitations, as follows. First and foremost, we do not explore *in detail* the entire space of design possibilities spanned by the normalization and selection of input features, possible distance metrics, classification techniques, and hyperparameters. This is, we believe, unavoidable, since this space is simply too large to densely sample along all its dimensions in an effective way. Nevertheless, we argue that the visual feedback provided by VA, via the different views of *featured* (observation, scoring, and features), coupled with the user's ability of directly controlling all aspects of the classification pipeline from within the tool, provides insights that allow the designer to use his/her intuition to limit the search effort toward finding a good design. We follow here the same rationale used earlier when coupling scientific visualization with numerical computation in so-called computational steering approaches [37]. Second, the ability of projections to accurately expose high-dimensional data structure is well known to be imperfect [33]. However, we do not use projections to predict actual classifier accuracy, but only to gain insights on general trends, such as the correlation of clusters with specific features and feature values, which next help our classifier engineering decisions.

Implementation *featured* is implemented mainly in Python, using Qt for the graphics interface. Classifiers, feature scoring techniques, and the t-SNE projection are provided via the *scipy*, *scikit-learn*, and *mlpy* Python packages [2, 23, 42]. Third-party projection techniques such as LAMP, IDMAP, and Sammon mapping, and LSP, are provided by the Java-based Projection Explorer framework [40] via Python wrapping. For GMLVQ, we based our implementation on the open-source code available at [7].

6 Conclusions

We foresee two types of effective extensions of this work, as follows. On the *technical* side, we aim to extend *featured* with mechanisms that provide a *consensus* outcome for its key dimensions (projections, feature scoring metrics, and classification techniques). This way, users can decide much easier on the importance of an obtained insight, e.g., based on a voting scheme. On the *application* side, we

aim to perform a more in-depth study of the prediction accuracy of prostate cancer relapse, based on more samples (patients), considering more dimensions (features), and studying how the machine predictions match predictions performed by actual medical specialists.

References

1. Abernethy, A.P., Etheredge, L.M., Ganz, P.A., Wallace, P., German, R.R., Neti, C., Bach, P.B., Murphy, S.B.: Rapid-learning system for cancer care. J. Clin. Oncol. **28**(27), 4268–4274 (2010). PMID: 20585094; https://doi.org/10.1200/JCO.2010.28.5478
2. Albanese, D., Visintainer, R., Merler, S.: mlpy: Machine learning Python (2012). arXiv:1202.6548; http://mlpy.sourceforge.net
3. Altman, N.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**(3), 175–185 (1992)
4. Bartenhagen, C., Klein, H.U., Ruckert, C., Jiang, X., Dugas, M.: Comparative study of unsupervised dimension reduction techniques for the visualization of microarray gene expression data. BMC Bioinform. **11**, 567 (2010). https://doi.org/10.1186/1471-2105-11-567
5. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In: Neural Networks: Tricks of the Trade, pp. 437–478. Springer, Berlin (2012)
6. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Thiel, K., Wiswedel, B.: KNIME – the Konstanz information miner: version 2.0 and beyond. ACM SIGKDD Explor. Newsl. **11**(1), 26–31 (2009)
7. Biehl, M.: GMLVQ source code. http://www.cs.rug.nl/~biehl/gmlvq (2017)
8. Boser, B., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: Proceedings of the 5th Annual Workshop on Computational Learning Theory, pp. 144–152. ACM, New York (1992)
9. da Silva, R.R.O., Rauber, P., Martins, R.M., Minghim, R., Telea, A.: Attribute-based visual explanation of multidimensional projections. In: Proceedings of EuroVis Workshop on Visual Analytics (EuroVA), pp. 137–142 (2015)
10. Demsar, J., Leban, G., Zupan, B.: FreeViz – an intelligent multivariate visualization approach to explorative analysis of biomedical data. J. Biomed. Inform. **40**(6), 661–671 (2007)
11. Domingos, P.: A few useful things to know about machine learning. Commun. ACM **10**(55), 78–87 (2012)
12. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. **63**(1), 3–42 (2006)
13. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
14. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**(1–3), 389–422 (2002)
15. Hajian-Tilaki, K.: Receiver operating characteristic (ROC) curve analysis for medical diagnostic test evaluation. Casp. J. Intern. Med. **4**(2), 627–635 (2013). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3755824/
16. Hammer, B., Villmann, T.: Generalized relevance learning vector quantization. Neural Netw. **15**, 1059–1068 (2002)
17. Hoffman, P., Grinstein, G., Marx, K., Grosse, I., Stanley, E.: DNA visual and analytic data mining. In: Proceedings of the IEEE Visualization, pp. 437–445 (1997)
18. Hofmann, M., Klinkenberg, R.: RapidMiner: Data Mining Use Cases and Business Analytics Applications. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, Boca Raton (2013)

19. Hohman, F., Kahng, M., Pienta, R., Chau, D.H.: Visual analytics in deep learning: an interrogative survey for the next frontiers (2018). arXiv:1801.06889 [cs.HC]
20. Hua, K.L., Hsu, C.H., Hidayati, S.C., Cheng, W.H., Chen, Y.J.: Computer-aided classification of lung nodules on computed tomography images via deep learning technique. OncoTargets Ther. **8**, 2015–2022 (2015). https://doi.org/10.2147/OTT.S80733; https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4531007/
21. Joia, P., Coimbra, D., Cuminato, J.A., Paulovich, F.V., Nonato, L.G.: Local affine multidimensional projection. IEEE Trans. Vis. Comput. Graph. **17**(12), 2563–2571 (2011)
22. Jolliffe, I.T.: Principal Component Analysis. Springer, Berlin (2002)
23. Jones, E., Oliphant, T., Peterson, P.: SciPy: open source scientific tools for Python (2017). http://www.scipy.org
24. Keim, D., Andrienko, G., Fekete, J.D., Görg, C., Kohlhammer, J., Melancon, G.: Visual analytics: definition, process, and challenges. In: Information Visualization – Human-Centered Issues and Perspectives, pp. 154–175. Springer, Berlin (2008)
25. Keim, D.A., Mansmann, F., Schneidewind, J., Thomas, J., Ziegler, H.: Visual analytics: scope and challenges. In: Visual Data Mining, pp. 76–90. Springer, Berlin (2008)
26. Kimelfeld, B., Ré, C.: A relational framework for classifier engineering. In: Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS '17, pp. 5–20. ACM, New York (2017). http://doi.acm.org/10.1145/3034786.3034797
27. Kohonen, T.: Learning vector quantization. In: Arbib, M. (ed.) The Handbook of Brain Theory and Neural Networks, pp. 537–540. MIT Press, Cambridge (1995)
28. Leban, G., Zupan, B., Vidmar, G., Bratko, I.: VizRank: data visualization guided by machine learning. Data Min. Knowl. Disc. **13**(2), 119–136 (2006)
29. Leemput, K.V., Maes, F., Vandermeulen, D., Suetens, P.: Automated model-based tissue classification of mr images of the brain. IEEE Trans. Med. Imaging **18**(10), 897–908 (1999). https://doi.org/10.1109/42.811270
30. Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J.Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D.M., Teichman, A., Werling, M., Thrun, S.: Towards fully autonomous driving: systems and algorithms. In: Intelligent Vehicles Symposium, pp. 163–168. IEEE, Piscataway (2011)
31. Liu, S., Bremer, P.T., Pascucci, V.: Distortion-guided structure-driven interactive exploration of high-dimensional data. Comput. Graph. Forum **33**(3), 101–110 (2014)
32. Liu, S., Maljovec, D., Wang, B., Bremer, P.T., Pascucci, V.: Visualizing high-dimensional data: advances in the past decade. IEEE Trans. Vis. Comput. Graph. **23**(3), 1249–1268 (2017)
33. Martins, R., Coimbra, D., Minghim, R., Telea, A.: Visual analysis of dimensionality reduction quality for parameterized projections. Comput. Graph. **41**, 26–42 (2014)
34. Meinshausen, N., Bühlmann, P.: Stability selection. J. R. Stat. Soc. **72**(4), 417–473 (2010)
35. Minghim, R., Paulovich, F.V., Lopes, A.A.: Content-based text mapping using multidimensional projections for exploration of document collections. In: Visualization and Data Analysis (Proceedings of SPIE-IS&T Electronic Imaging), vol. 60, pp. 606–615 (2006)
36. Mühlbacher, T., Piringer, H., Gratzl, S., Sedlmair, M., Streit, M.: Opening the black box: strategies for increased user involvement in existing algorithm implementations. IEEE Trans. Vis. Comput. Graph. **20**(12), 1643–1652 (2014)
37. Mulder, J., van Wijk, J.J., van Liere, R.: A survey of computational steering environments. Futur. Gener. Comput. Syst. **15**(1), 119–129 (1999)
38. Niknazar, P., Bourgault, M.: In the eye of the beholder: opening the black box of the classification process and demystifying classification criteria selection. Int. J. Manag. Proj. Bus. **10**(2), 346–369 (2017)
39. Paller, C.J., Antonarakis, E.S.: Management of biochemically recurrent prostate cancer after local therapy: evolving standards of care and new directions. Clin. Adv. Hematol. Oncol. **11**(1), 14–23 (2013)
40. Paulovich, F., Oliveira, M.C.F., Minghim, R.: The projection explorer: a flexible tool for projection-based multidimensional visualization. In: Proceedings of SIBGRAPI, pp. 27–36 (2007)

41. Paulovich, F., Nonato, L., Minghim, R., Levkowitz, H.: Least square projection: a fast high-precision multidimensional projection technique and its application to document mapping. IEEE Trans. Vis. Comput. Graph. **14**(3), 564–575 (2008)
42. Pedregosa, F., Varoquaux, G., Gramfort, A., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011). http://scikit-learn.org
43. Pennacchiotti, M., Popescu, A.M.: A machine learning approach to twitter user classification. In: ICWSM, vol. 11, pp. 281–288 (2011)
44. Pezzotti, N., Höllt, T., van Gemert, J., Lelieveldt, B.P., Eisemann, E., Vilanova, A.: DeepEyes: progressive visual analytics for designing deep neural networks. IEEE Trans. Vis. Comput. Graph. **24**(1), 98–108 (2018)
45. Rauber, P., da Silva, R., Feringa, S., Celebi, M., Falcão, A., Telea, A.: Interactive image feature selection aided by dimensionality reduction. In: Proceedings of EuroVA, pp. 46–51. Eurographics (2015)
46. Rauber, P., Fadel, S., Falcão, A., Telea, A.: Visualizing the hidden activity of artificial neural networks. IEEE Trans. Vis. Comput. Graph. **23**(1), 101–110 (2017)
47. Sammon, J.W.: A non-linear mapping for data structure analysis. IEEE Trans. Comput. **C-18**, 401–409 (1964)
48. Shen, D., Wu, G., Suk, H.I.: Deep learning in medical image analysis. Ann. Rev. Biomed. Eng. **19**(1), 221–248 (2017). http://dx.doi.org/10.1146/annurev-bioeng-071516-044442
49. Sorzano, C., Vargas, J., Pascual-Montano, A.: A survey of dimensionality reduction techniques (2014). http://arxiv.org/pdf/1403.2877
50. Stephenson, A.J., Kattan, M.W., Eastham, J.A., Dotan, Z.A., Bianco, F.J., Lilja, H., Scardino, P.T.: Defining biochemical recurrence of prostate cancer after radical prostatectomy: a proposal for a standardized definition. J. Clin. Oncol. **24**(24), 3973–3978 (2006)
51. Sun, Y.: Iterative relief for feature weighting: algorithms, theories, and applications. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1035–1051 (2007)
52. Talbot, J., Lee, B., Kapoor, A., Tan, D.: EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers. In: Proceedings of ACM CHI, pp. 1283–1292 (2009)
53. Tamagnini, P., Krause, J., Dasgupta, A., Bertini, E.: Interpreting black-box classifiers using instance-level visual explanations. In: Proceedings of ACM HILDA (2017)
54. van der Maaten, L.: Learning a parametric embedding by preserving local structure. In: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS) (2009)
55. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2431–2456 (2008)
56. van der Maaten, L., Postma, E., van den Herik, H.: Dimensionality reduction: a comparative review. J. Mach. Learn. Res. **10**(1), 66–71 (2009). http://www.iai.uni-bonn.de/~jz/dimensionality_reduction_a_comparative_review.pdf
57. Zhang, J., Gruenwald, L.: Opening the black box of feature extraction: incorporating visualization into high-dimensional data mining processes. In: Proceedings of IEEE International Conference on Data Mining (ICDM) (2006)