Abstract

Recently introduced visualization tools such as Silhouettes, Class Maps, and Quasi-residual plots, available via the CRAN-released "classmap" R package, provide valuable insights into the workings and effectiveness of ML algorithms in a specific classification tasks, as well as into the data itself. This paper comprehensively reviews these graphical displays and introduces a new one, named MDScolorscale, to the existing toolbox. A detailed explanation on how to manually produce the necessary quantities for these visualizations is provided, together with a versatile function that enables access to the graphical displays for classifications made by virtually any ML classifier beyond the currently supported ones. These tools are then further illustrated by applying them to a gene-expression-based cancer classification problem processed by the Nearest Shrunken Centroid Classifier (also known as "pamr" in its microarray application) and the XGBoost classifier. Specific support for the visualizations is developed for the R "pamr" ML classifier. The added features to the original package can be accessed by loading an auxiliary package, available at https://github.com/LLazzar/classmapExt.

Keywords: Visualization tools, Classification Algorithms, Silhoutte, Quasi-residual plots, PAMR

Contents

1 Introduction

As we navigate through the era of big data, Machine Learning (ML) algorithms have emerged as potent tools for extracting meaningful information from complex data structures, finding application across diverse fields such as finance, retail, and marketing. Here we focus on a particular subset of these applications: classification tasks, where ML algorithms are employed to predict discrete target variables based on a set of features. Examples of such tasks range from credit risk assessment in banking to tumor detection in medical imaging.

Understanding the performance of classification algorithms in relation to specific datasets they are applied to is crucial. Gaining insights into the strengths and weaknesses of a model with regard to a given problem serves as a diagnostic tool that guides model improvement and fine-tuning. Moreover, such understanding fosters conscious and informed deployment in practice. Within this framework, graphical visualization tools prove to be particularly effective aids.

Research shows that the human brain often processes visual data more effectively than numerical data. As such, graphical tools can help reveal patterns and structures within data that might remain obscured under numerical examination. These tools also serve as invaluable resources for translating intricate ML concepts into more readily comprehensible information, particularly when communicating results to individuals outside the field or those with less technical expertise. However, as ML algorithms continue to evolve and grow in complexity, there's a concurrent need for diagnostic graphical tools to advance and offer fresh perspectives to adequately represent and interpret the outcomes.

This paper advances the aforementioned progress by providing a thorough review and diffusion of a recently introduced set of visualization tools, devised by Raymaekers and Rousseeuw. These tools, namely "Silhouette plots," "Quasi-residual plots," and "Classmaps," center on the analysis of classified cases for which the true class membership is known. These can be used on both training and independent test sets and are accessible through the R-CRAN released 'classmap' package, which offers support for a variety of popular ML classifiers.

In Section 2, these visualization tools are detailed with clarity, revealing the underlying statistical theory and the essential quantities that need to be computed. Furthermore, an additional graphical tool called the "MDS color-scale plot" is introduced to broaden the existing toolkit.

Section 3 delves into the specifics of how to access these visualizations within the R language framework. This includes a comprehensive overview of the 'classmap' package, which houses the original visualization tools. In addition, a new, versatile function is proposed that offers adaptive support for accessing visualizations, extending beyond the pre-defined supported algorithms. Finally, the 'classmapExt' package, introducing newly added functionalities, is discussed.

In response to the growing use of ML classifiers in the biotech field, Section 4 scrutinizes a cancer classification problem based on gene expression data, a realm where ML classifiers have increasingly demonstrated their utility. A well-known classifier in the literature, the Nearest Shrunken Centroid (NSC) - often referred to as "pamr" in its application to microarray gene-expression data - is employed for this task, specifically on a problem involving a novel type of gene expression data. Specific support for the visualizations is developed for the R "pamr" algorithm and utilized in this analysis. Moreover an XGBoost classifier is also employed. Since XGBoost is not among the algorithms with explicit support for the visualizations, some quantities are manually calculated and the newly presented flexible function is engaged to access the graphical tools. The utility of the proposed visualizations is demonstrated in the data analysis, shining a light on results for the specific classifiers and offering a comparison between them, thus providing a practical exemplification of their application.

2 The novel graphical displays

In 2022, J. Raymaekers and P.J. Rousseeuw introduced a novel set of graphical displays that proved to bring valuable insights for ML classification tasks (M. H. Jakob Raymaekers P. J. R. 2022) (P. J. R. Jakob Raymaekers 2022). Such displays can be applied to labeled cases within

the training or testing sets that have been processed by the ML classifier under consideration. They provide, in an intuitive and immediate way, visual benchmarks of the accuracy of predictions for individual cases and entire classes, as well as the evaluation of factors that may impact the quality level of those predictions. Examining the produced plots can reveal patterns and structures that are key to understanding several aspects of the data processing.

Before discussing the graphical displays, it is necessary to define the simple statistical entity they are based on, referred to as the Probability of Alternative Class (PAC).

2.1 Core Statistical Concept: PAC

A generic ML classifier algorithm is considered along with a set of n labeled observations indexed by $i = 1 \ldots n$. This set is typically the training set on which the ML algorithm has been trained, or an independent test set used for evaluation. Additionally, a discrete set of g classes/labels with elements indexed by $g = 1 \ldots .G$ is defined, and used by the algorithm during the training process. The class to which each observation i belongs is known and denoted by g_i.

We assume that once the algorithm is given an observation i, it can generate a discrete posterior probability distribution $\hat{P}_i(g)$ that describes the probability of the observation belonging to each class g. This distribution may be produced naturally by the algorithm's design or may be derived coherently following the algorithm logic. In either case, the algorithm should predict the class to which the observation i with $\hat{g}_i$, where $\hat{g}_i = \underset{g \in G}{\arg\max}\, \hat{P}_i(g)$.

Now we evaluate the maximum of the posterior probability distribution achieved by a class different than the true one $g(i)$:

$$\tilde{P}_i = \max \left\{ \hat{P}_i(g) \, ; \, g \neq g_i \right\}$$

and this happens for the class $\tilde{g}_i$:

$$\tilde{g}_i = \underset{g \in G; g \neq g_i}{\arg\max}\, \hat{P}_i(g)$$

$\tilde{g}_i$ can be interpreted as the best prediction in the eyes of the classifier if the true g_i wasn't a viable option. We call that the "Best Alternative Class". Now we can state that:

- If $\hat{P}_i(g_i) > \tilde{P}_i = \hat{P}_i(\tilde{g}_i)$ the classifier assign the observation to the true class g_i (1)
- If $\hat{P}_i(g_i) < \tilde{P}_i = \hat{P}_i(\tilde{g}_i)$ the classifier assign the observation to the wrong class $\tilde{g}_i$ (2)

Now we have defined all the quantities needed to construct the Probability of the Alternative Class (PAC) for the generic *i*-th observation:

$$PAC(i) = \frac{\tilde{P}_i}{\hat{P}_i(g_i) + \tilde{P}_i} \text{ , with } PAC(i) \in [0,1]$$

$PAC(i)$ is derived as a conditional probability, and like a proper probability measure it ranges between 0 and 1. It represents, in relation to the classifier, the probability of assigning i to the "Best Alternative Class", given either the probability of assign i to the true class or the "Best Alternative Class" itself. A meaningful threshold is implicitly defined as a consequence of (1) (2):

- If $PAC(i) < 0.5$ the classifier assigns the observation to the true class g_i
- If $PAC(i) > 0.5$ the classifier assigns the observation to the wrong class $\tilde{g}_i$

By relating the posterior probability of the best alternative class to that of the true class, *PAC* provides a continuous quantitative estimate of the classifier's level of agreement with the true class. It is important to observe that in devising this measure, only the "gap" between the $\tilde{g}_i$ class and the g_i true class is considered, among all the information in the posterior probability space.

A *PAC* near 0 indicates that the classifier is strongly convinced in predicting the correct class, while a *PAC* near 1 means that the classifier is not convinced at all in giving the correct prediction and almost surely prefers $\tilde{g}_i$. Thus, in essence, the *PAC* value of a given observation i, can be considered as a proxy that, in a scale between 0 and 1, gives the degree of confidence that the considered classifier has in predicting the correct class.

2.2 Silhouette Plot for classification

The silhouette plot was initially proposed in the context of clustering (Rousseeuw 1987) and has since gained widespread use. Its main purpose is to quantify how well an observation fits into its assigned cluster, given a particular clustering algorithm, thereby serving as a robust tool for evaluating the resulting groupings. Leveraging its core principles and inherent logic, the silhouette plot has been adeptly retooled for application in the classification context. In this setting, it is used to quantify how well an observation fits in its true class g_i for the once-trained ML classifier under consideration.

To compute the silhouette width $s(i)$ for each observation, we leverage the information of the PAC value, reverse its direction and stretch it to a range of -1 to 1. Specifically:

$$s(i) = 1 - 2PAC(i)$$

This definition allows us to obtain a silhouette width that shares the same properties and interpretation as the one used in clustering. Values close to 1 indicate a very good fit for the observation in the true class for the classifier and values close to -1 indicate an extremely poor one. Now higher values means a higher conviction in predicting the correct class and interpretation is more straightforward.

And now the significant threshold is set at 0:

- If $s(i) > 0$ the classifier assigns the observation to the true class g_i

- If $s(i) < 0$ the classifier assigns the observation to the wrong class $\tilde{g}_i$

To generate the silhouette plot, a set of observations is considered, typically the entire training or testing set, and their relative silhouette width values are plotted as lines along the x-axis. These are organized in a stacked vertical manner, grouped by the true class (and colored accordingly) to which they belong. Within each class, they are sorted in descending order based on the values they take. The silhouette plot also includes the overall average silhouette width for each class, and overall, to facilitate numerical evaluation.nOn the right side of Figure 1, the silhouette plot is drawn, based on the results of a 4-class classification on a certain set of observations with known labels. On the left side, a confusion matrix pertaining to the same case is drawn.

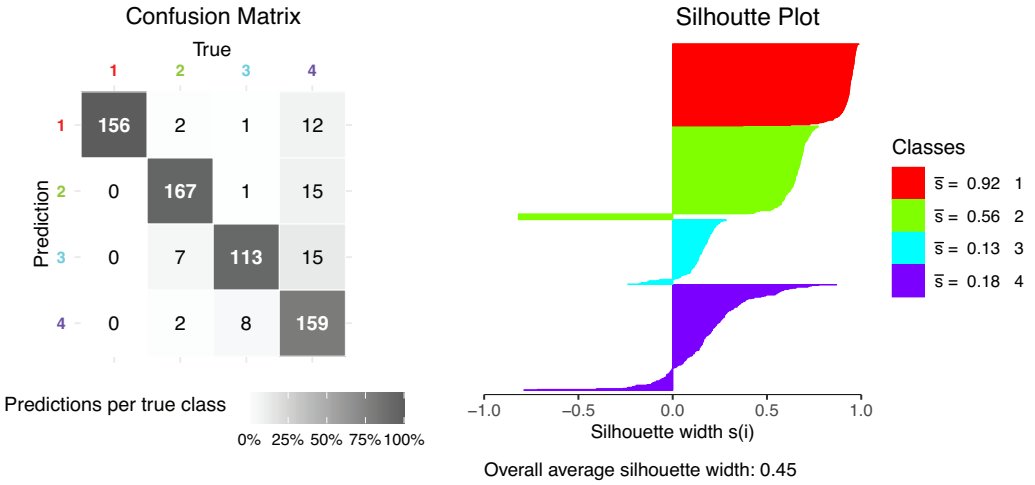

Figure 1: The Confusion Matrix on the left and the Silhoutte Plot on the right. They display the results of the same 4-class classification task on a set where true classes are known.

Confusion matrices and derived metrics such as recall and precision are perhaps the most used tools in evaluating classification. A confusion matrix can provide instant information about misclassification and which classes are being confused with others, thereby providing important information for classification. However, they are rather strict in their binary distinction between misclassified and correctly classified instances, neglecting some of the information contained in the posterior probability of the classification.

Scrutinizing the posteriors probabilities numerically or finding a convenient, uncluttered way to display them is not an easy task. This is where the silhouette plot comes in, complement-

ing information from displays like a confusion matrix, but going beyond the boundaries of classified/not classified and providing a glance at the degree of confidence toward the correct prediction.

Referring back to the case represented in Figure 1, an examination of the confusion matrix shows that the predictions for class 2 and 3 appear rather equivalent; the classes are generally predicted accurately except for 10 and 11 misclassifications each. However, a closer look at the silhouette plot reveals considerable differences in classifier performance. Instances correctly classified in class 2 exhibit strong prediction confidence, while the misclassified instances are noticeably off the mark. Conversely, for class 3, correctly classified cases are accurately predicted but without substantial confidence, and misclassified cases are only slightly misjudged. Class 1 instances are correctly predicted with high confidence, boasting an average silhouette score of 0.92. Lastly, class 4 shows overall prediction quality that is somewhat lacking, with many misclassifications, some of which are severely incorrect. Correct predictions are not particularly strong, with a few exceptions.

2.3 Quasi Residual Plot

A quasi-residual plot is a scatter plot that represents PAC values on the Y-axis, and a selected data feature on the X-axis. The chosen feature may or may not have been involved in the model's training. For ease of interpretation, the plot area is divided into two distinct colored sections. The white section denotes points with PAC values exceeding 0.5, which contain misclassified instances. Conversely, the gray section represents points with PAC values below 0.5, signifying correctly classified instances.

This tool can be used to assess whether changes in the values of the considered feature have an impact on the accuracy of the classifier, as measured by the PAC. Multiple curve types, such as average plus-minus standard error, a loess curve, among others, can be overlaid to discern PAC value patterns and trends.

Quasi residual plots are specifically tailored for continuous features, but they can also be used to study PAC values for discrete covariates.

In Figure 2, two examples of quasi-residual plots are illustrated. It appears that different values for feature 2 do not substantially influence the classifier's precision, which, on average, registers a PAC just above 0.2, regardless of changes in feature values. However, this is not the case for feature 1. High values of this variable show a considerably strong positive correlation with PAC values. As values in feature 1 increase, the classifier's prediction accuracy tends to deteriorate.

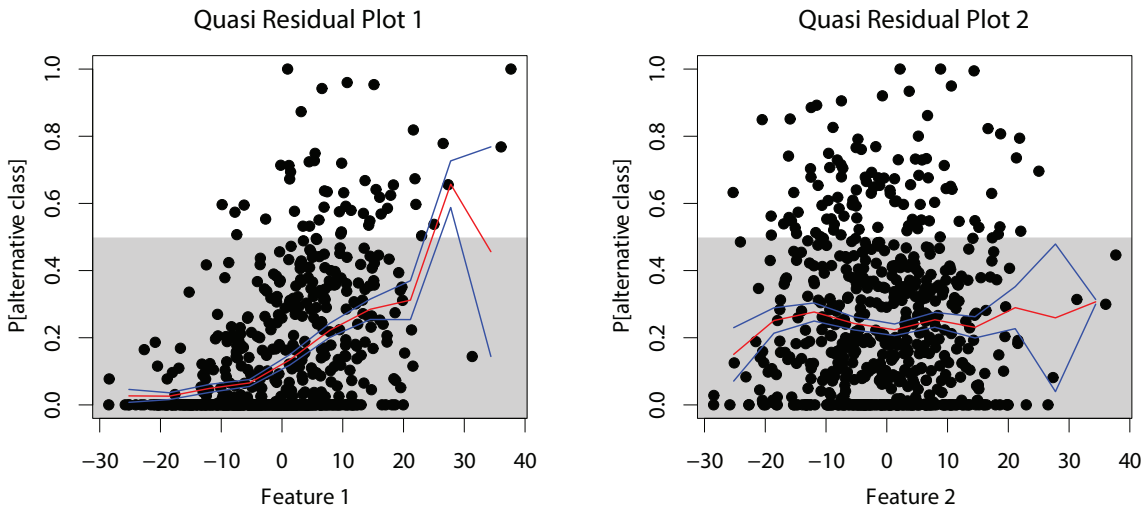

Figure 2: Two examples of quasi-residual plots with average PAC values in red and average plus/minus standard error in blue.

Evaluating these plots can aid in understanding and controlling covariates external to the model, and deciding whether it may be worth incorporating them. Even when certain information cannot be integrated, it's beneficial to identify if a particular covariate poses a challenge to the model's accuracy (and this is to keep in mind for example when deploy the model). These plots further enhance the understanding of the classifier's strengths and weaknesses in relation to individual data features, including those already part of the model.

Evaluating these plots can enhance understanding and management of covariates not present in the model, thereby guiding decisions regarding their potential integration. Even when informa-

tion within a covariate cannot be incorporated, it's still crucial to determine if it may adversely affect the model's accuracy, and be aware of any potential pitfalls when the model is being utilized. So in summary quasi-residual plots may improve the comprehension of the classifier's strengths and weaknesses in relation to individual data features, including those already part of the model

It is important to note, given that this plot doesn't distinguish between classes, that the emergence of a significant correlation between a certain feature and the PAC doesn't necessarily indicate that the classifier's overall precision is significantly impacted when that feature changes. It might also be possible that the feature is correlated with the likelihood of observations belonging to a particular class, which the classifier predicts either better or worse than average. Lastly, the lack of correlation between PAC and a feature does not denote that the feature is irrelevant to the model.

2.4 Classmap Plot

A classmap is a specific type of quasi residual plot. It considers only a particular class and displays the observations within that class according to the true known labeling. Moreover, the X-axis represents a derived metric called $farness$ for each object. This metric is devised as a proxy for the distance an observation has from its true class through the lens of the classifier.

In relation to the specific ML algorithm employed for the analyzed classification, a distance-type measure $D(i,g)$ is carefully specified to calculate how far an observation i is from a class g. This is done by considering how the once-trained classifier operates and perceives data in defining its decision space. The definition of the measure resides, more or less explicitly, in the classifier algorithm itself and in what it learned fitting the data at hand. This includes its method of establishing class prototypes during the training process, where it makes use of the true labels, and subsequently how each observation is processed for prediction afterwards.

This metric enables the calculation of a value $D(i,g_i)$ for each i observation, which indicates its

distance from its true class g_i. Subsequently, $D(x, g_i)$ is introduced. This quantity represents the distance of an x random generated observation belonging to class g_i to the originating class g_i itself. Starting from the set of the calculated values $D(i, g_i)$ available, the cumulative distribution function of $D(x, g_i)$ is aptly estimated and referred to as $\hat{F}$. For detailed steps on this estimation process please refer to Section A.1 of the supplementary material in [7]. The 'farness' of the i-th observation to its class g_i is defined as a probability:

$$farness(i) = P[D(x, g_i) \leq D(i, g_i)] = \hat{F}[D(i, g_i)], \text{ with } farness(i) \in [0, 1]$$

Where P would be the probability distribution of $D(x, g_i)$. $\hat{F}$ is estimated by taking into account the collection of $D(i, g_i)$ values from the training set. If the $farness$ value for an observation inside a test set needs to be calculated, the previously estimated $\hat{F}$ from training should be used.

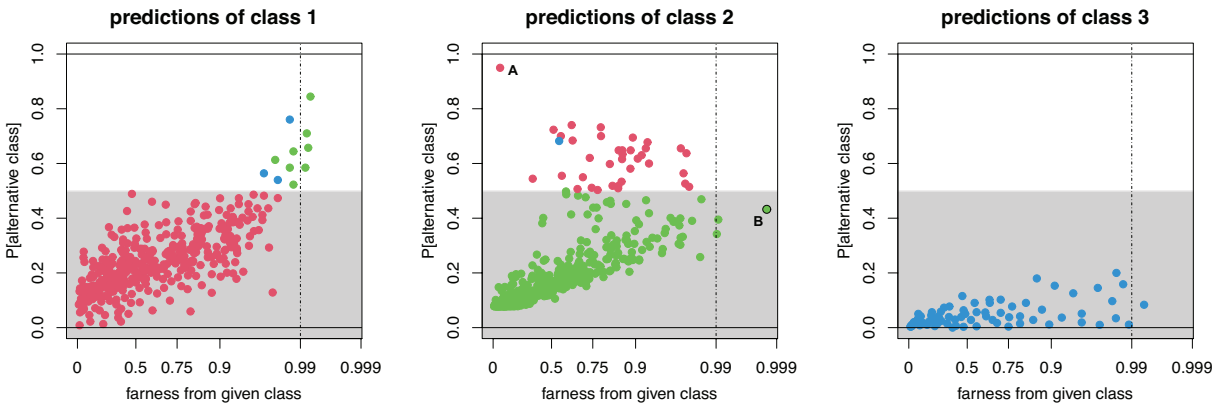

Figure 3: Classmaps on the results of a 3-class classification problem

Figure 3 depicts classmaps for a 3-class classification problem. Most observations in class 1 are correctly classified, with accuracy gradually dropping as farness increases. A small group composed of the farthest observations is misclassified, primarily for class 3, represented in green. In fact, the plot's color-coded points, based on predicted class, provide clarity on misclassifications. Class 2 shows significant misclassification for class 1, suggesting class overlap within the classifier's decision space. Misclassified point A, very near class 2, warrants further

investigation and potential mislabeling, provided that's plausible for the dataset under review. Conversely, point B, despite being far from class 2, is correctly classified with a near 0.5 PAC value, and it is marked as an outlier with a black border. An outlier in a class map plot is a point that, considering the set of all farness values to all possible classes and not only the canonical one to its true class, achieves a minimum value of at least 0.99. This means that the point is not only very far from its assigned class, but also from all the other classes. Lastly, class 3 observations maintain high prediction accuracy even at increased distance.

Classmaps can shed light on how a classifier's PAC-measured accuracy can be affected when moving away from its optimal class portrayal. They provide valuable information on the destination class of mislabels, showcasing potential overlaps and relationships among classes. They can aid in identifying troublesome instances for the classifier, which can then be examined in more detail to further understand the classifier's viewpoint on the task.

2.5 MDS Color-scaled Plot

The MDS color-scaled plot is a novel addition introduced in this paper. It enriches the existing repertoire constituted by the previously reviewed graphical visualizations, all of which focus on conveying PAC-measured precision for a classification in a variety of ways. This new graphical display leverages PAC information inside a two-dimensional spatial representation of the classified observations.

For the given set of observations for which the result of a classification wants to be analyzed, all pairwise distances between those observations need to be computed. This computation calls for a distance-type metric $D(r,s)$ to be devised, with r and s representing generic observations from the considered set. This metric should be crafted coherently with the perspective that the employed trained classifier has about the data in its formed decision space. This is similar to what we do for the computation of $farness$ in the classmap plot, where we evaluate the distance between a point and each class instead of the distance between two points.

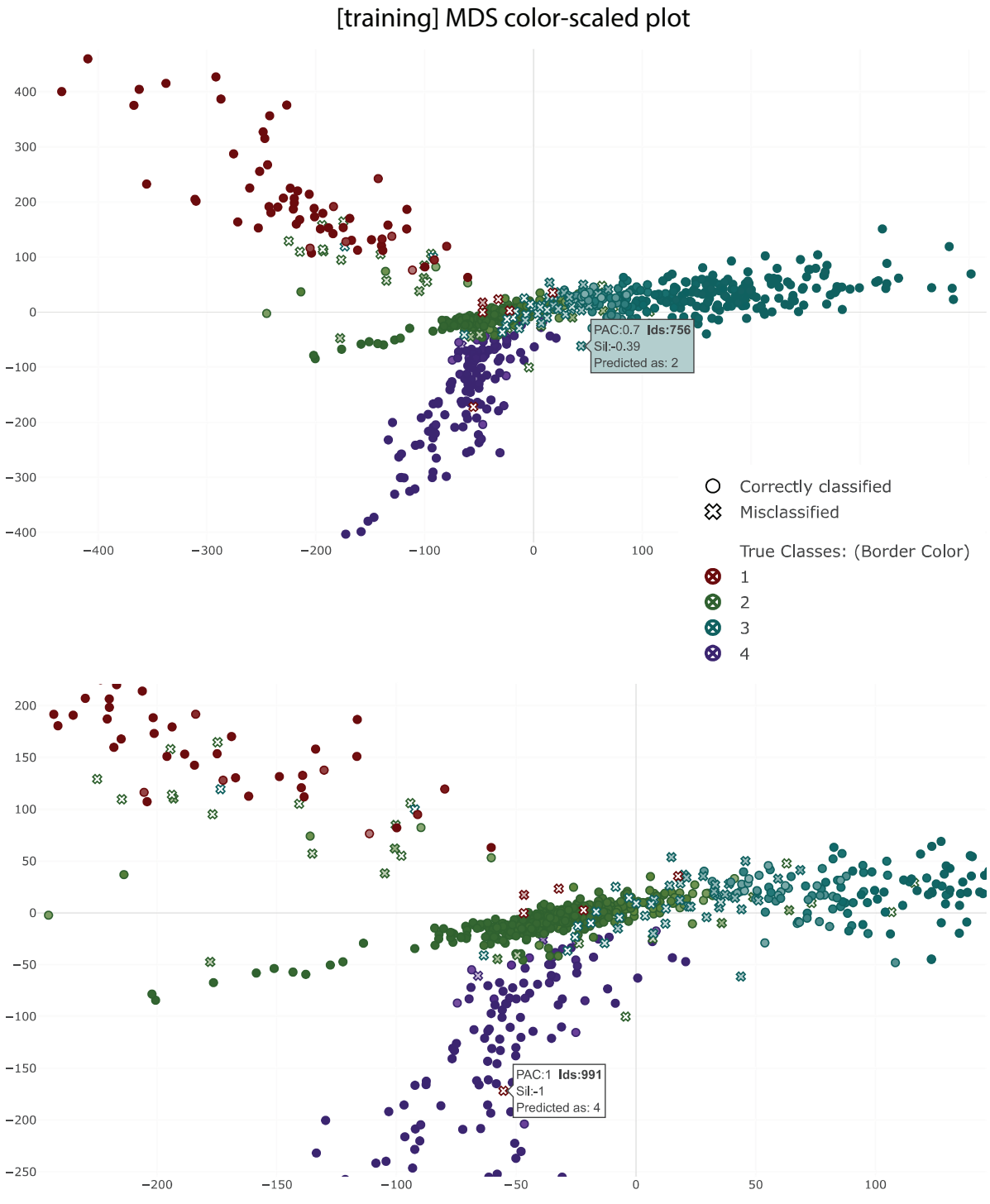

Figure 4: The MDS-color scaled plot (above) displays the results of a four-class classification problem on the training set, with a zoomed-in view (below).

This collection of pairwise dissimilarities is then utilized to organize the observations in a two-dimensional space in a distance-wise fashion, accomplished via multidimensional scaling

(MDS). Each observation is plotted with a color associated with the class it truly belongs to. The color, in its unaltered form, is used for the border of the point, while for the fill its intensity is mapped to the PAC (Sil) value. A less intense fill, tending towards white, indicates higher PAC values (lower Sil) and diminished confidence in correctly classifying the observation represented by the point. Observations with PAC values below 0.5, which are misclassified, are drawn with an 'X' shape. In this way, we succeed in incorporating the PAC-measured accuracy of the classification into a coherent spatial visualization of the points.

The visualization enhances the comprehension of the classifier's decision boundaries, both in terms of their position and their gradual transition, within a space that uses color to differentiate between the true classes. It is possible to comprehend the view of the trained ML algorithm on the cases it is classifying about the relationship and degree of separation between classes, as well as their relative spread or compactness of their composition.

In Figure 4, a MDS color-scaled plot is presented for the results of 4-class classification on the training set. The figure also includes a zoomed-in view of the main cloud of data points. The visualization, generated through the plotly library, is designed to be interactive for improved navigation and focus on distinct point regions. Hovering on a particular observation unveils its IDs, denoting its location in the data matrix, the specific PAC/SIL values it scores, and the class it's predicted to belong to in the classification.

In the illustrated case, classes 2 and 3 are grouped closely with a certain overlap, leading to suboptimal decision boundaries and frequent class 3 misclassifications as class 2. Class 4 is near class 2, yet their boundaries are clearer and more strongly defined (no significant PAC increase) in the 2D projection. Many misclassified points from class 2 are near class 1's prototype, leading to misclassification for class 1. The anomalous point 991 from class 1, located within the core area of class 4, is misclassified. Further examination might be necessary to comprehend why this occurs. Better discrimination exists among classes 1, 3, and 4, while class 2, excluding a densely populated subgroup, tends to infringe the representations of the other classes.

3 Implementation

The functions to create the three reviewed graphical displays, namely the silhouette, quasi-residual plot, and classmap, are provided in the R package 'classmap' (P. R. Jakob Raymaekers 2023), released on CRAN, where Raymaekers and Rousseeuw have meticulously implemented their proposals. The respective function names are 'silplot', 'qresplot', and 'classmap'. These are not immediately ready-to-use with general results of a given classification. This is because for the specific results and ML algorithm employed, some processing is needed to obtain the particular required quantities for each classified case, which are the *PAC* and the $farness$ value.

To compute the *PAC*, it is necessary to obtain the posterior probabilities pertaining the considered classification. When working with classifiers like KNN, which do not inherently provide posterior probabilities, one must employ a suitable technique tailored to the algorithm in order to derive coherently these estimates.

To compute the $farness$, as explained in section 2.4, the distance-type measure $D(i,g)$ should be devised specifically for the classifier under consideration. Following this, a general procedure can be applied to estimate the cumulative distribution function required to determine the $farness$ value for each observation.

Currently, the classmap package provides support for six distinct machine learning algorithms, including Discriminant Analysis, Random Forest, K-Nearest Neighbor, Neural Networks, CART decision trees, and Support Vector Machines. For each of these algorithms, there are corresponding functions `vcr.*.train` and `vcr.*.test`, where `*` represents a chosen short name for the algorithm. It is crucial to note that there are separate functions for visualizing the training set and the test set results, and the output of `vcr.*.train` is always required as input for `vcr.*.test`. This requirement exists because it is essential to estimate posterior probabilities (when implicit definition is needed) and calculate the $farness$ value for the test set, taking into account only the specific observation and the information gathered from the training set (used for the estimation the cumulative distribution function $\hat{F}$). In other words, we cannot utilize

all the information in the test set collectively, as test observations are not meant to participate in the training process or, in our context, in constructing the quantities for the graphical displays. Then, the output of `vcr.*.train/test` can be fed into the specific plotting function, like `silplot`, to draw the desired visualization.

The vcr.*.* functions available for visualization preparation are constrained by the limited range of supported algorithm. They can also be considered somewhat rigid, as they either directly execute the algorithm by calling a specific R package themselves or require the output of a specific R function from a package as input. For example, `vcr.forest.*` requires the output of the RandomForest function within the randomForest package, and `vcr.svm.*` for Support Vector Machine requires the fit of the specific `svm` function in the `1071` package.

This rigidity does not offer an immediate method for researchers to use the plotting functions within the classmap package framework with their algorithms of choice. For instance, a researcher may have developed a nuanced, custom ML algorithm based on one of the baseline algorithms or employed another one not included in the list of six. They might also have chosen a different framework to develop and train their algorithm, such as a Python-based environment. The only truly flexible function is the one defined for Neural Networks Classifiers (NNC), called vcr.neural.train. This function requires only the coordinates of the n objects in the training data for a specific, arbitrarily chosen layer, the vector of posterior probabilities, and the true labels. The user can simply train the NNC using their preferred environment and custom methods, and then feed the results in the relative function to prepare for the visualizations.

In the spirit of expanding the flexibility and usability of the classmap package and its helpful visualizations, we have defined a function called `vcr.custom.*` which output can be used to access the plotting functions more broadly. With this function, users can input the posterior probabilities from the classification conducted by their selected algorithm. These probabilities can either be directly extracted from the output or manually constructed if not provided by the algorithm. Users should also provide the vector containing the true classes of the observations

within the classified set. This enables *PAC* and silhouette width computation. It is also given the option of inputting a matrix that contains the distance of each observation i to each class g, calculated using a personally devised measure $D(i,g)$. This, in turn, enables the calculation of $farness$. The function is accessible through the GitHub released package 'classmapExt', available at github.com/LLazzar/classmapExt.

The 'classmapExt' R package is designed to complement the 'classmap' R package, extending its functionality. Besides `vcr.custom.*`, it also includes the `mdscolorscale` function. This function takes the output of a `vcr.*.*` that processes a given classification and a matrix of dissimilarities, properly computed for each case, to produce the MDS Color-scaled Plot detailed in section 2.5. Moreover, it provides the `vcr.pamr.*` function that adds specific ad-hoc support to the visualizations for the R "pamr" ML classifier. This algorithm is applied in the classification problem scrutinized in section 4, where the development of the specific `vcr.pamr.*` function is also explained.

4 Visualizing Gene Expression-Based Cancer Classifications

The suite of visualizations presented are now applied to a specific classification task. This task has been selected from the rapidly evolving domain of biotechnology, where machine learning techniques are routinely deployed. The task is centered around differentiating various cancer subtypes based on gene-expression profiles derived from tissue samples.

This classification problem was specifically chosen because of the potential benefits offered by our visualizations. As these problems typically involve relatively small datasets, a meticulous and detailed analysis of both training and testing results is absolutely crucial. In such scenarios, our visual representations can provide valuable insights and help decipher complex nuances within the data.

Moreover, in this context, machine learning classifiers often serve a dual purpose. Beyond building a mere predictive model, they are employed to enrich our comprehension of the un-

derlying data and its inherent structure, including any substructures present. The proposed graphical visualizations contribute meaningfully to this objective, shedding light on the intricacies of the data while elucidating the relationships and trends therein. Thus, these graphical tools not only assist in accurate classification but also augment the broader understanding of the disease's genetic footprint.

4.1 Aspects of the task

Cancer classification based on gene expression analysis involves measuring the expression levels of thousands of genes in sampled tissue. This can be achieved through various techniques such as microarrays or next-generation sequencing (NGS) technologies such as RNA-seq. While microarrays are a mature and well-established technology, RNA-seq has gained popularity in recent years due to its ability to provide more comprehensive and accurate gene expression data. As a result, RNA-seq is gradually becoming the primary technique used to assess gene expression profiles in biological experiments (Bruno César Feltes and Dorn 2021).

After raw data is collected using a selected method, it undergoes a series of preprocessing steps that generally include quality checks, reference genome/transcriptome alignment, gene/transcript expression quantification, and normalization (Alonso-Betanzos et al. 2019). The normalization step is particularly critical, different statistical methods could be adopted in order to ensure between-sample comparability. Once preprocessed, the dataset can be fed into properly ML algorithms that are trained to identify, in their own style, the patterns that distinguish healthy tissue from different types or subtypes of cancerous tissue. These biological datasets that classifiers are trained on have particular characteristics that present challenges when working with this kind of data:

- **Small Sample Size**: Most gene expression datasets suffer from limited sample availability, partly due to cost and accessibility issues, and partly because the subjects involved are diseased individuals. Additionally, obtaining samples for rarer diseases and conditions can become increasingly difficult. The issue of sample scarcity poses a serious

threat of overfitting, as classifiers may learn to capture noise instead of the underlying biological patterns. It is crucial to implement adequate techniques and later validate the classifier's performance on independent datasets in order to ensure its generalizability to new samples. It is also important to be aware of potential dataset shift phenomena, which can occur when the distribution of the validation data differs from the training data.

- **Dimensionality and feature selection**: The high-dimensionality of genetic data, which is characterized by a large number of genes that must be considered, often requires the use of feature selection techniques to identify a subset of relevant genes for building a classifier. Different feature selection techniques may produce different results, thus, it is crucial to find discriminating subspaces that capture the most relevant biological information.

- **Noise and heterogeneity**: Noise can arise from both technical and biological sources, such as batch effects, sample quality, and inter-individual variability. This can pose a challenge in identifying robust biomarkers and developing reliable classifiers. Model stability should be taken into consideration and appropriate preprocessing techniques should be employed.

- **Class Imbalance**: Some cancer types may be more rare than others, leading to fewer samples available for certain classes. This can result in classifiers being biased towards the majority class. Several strategies, including resampling methods, cost-sensitive learning, and ensemble techniques, have been proposed to address class imbalance and improve classifier performance.

- **Model interpretability**: Understanding and interpreting the complex relationships between gene expression data and clinical outcomes is a major challenge in cancer classification. It is important to not only build accurate models but also understand the biological mechanisms underlying the classification. Additionally, exploring how various biological, clinical, and technical factors influence and impact algorithmic decisions is imperative.

- **Batch effects**: when merging datasets, very careful (Su Bin Lim and Lim 2018)

Despite these challenges, gene expression-based cancer classification has shown great promise in improving cancer diagnosis, prognosis, and treatment selection. It is an active area of research in cancer biology, and new algorithms and technologies are constantly being developed to improve the accuracy and robustness of these classifiers.

There are several advantages to using gene expression-based cancer classification. For example, it can provide a more objective and quantitative assessment of tumor characteristics compared to traditional histological methods. It can also identify molecular subtypes of cancer that may have different prognoses and treatment responses, which can ultimately lead to more personalized and effective treatments for patients.

4.2 Dataset

// WORK IN PROGRESS //

4.3 Methods

The task of classification is accomplished using the Nearest Shrunken Centroid (NSC) algorithm. This ML classifier, which was adapted from the original nearest centroid classifier (Tibshirani R. and G. 2002), is purpose-built for cancer class prediction from gene expression profiling. Named `pamr` (Prediction Analysis for Microarray) in its R language implementation, it was initially developed for microarray data (Tibshirani R. and G. 2003). Here we have decided to repurpose this method, testing its prowess on RNA-seq data. The NSC algorithm offers the advantage of integrated feature selection in its processing. This is a key step for handling the large data dimensions typical of gene expression datasets and helps prevent overfitting, diminishes noise capture, and avoids computational problems. Further, the algorithm preserves its operational capability to perform the classification task even when dealing with small sample sizes. To access the graphical displays, we have developed a function to specifically support a

`pamr` classification in R, taking into consideration the classifier's specific mechanisms.

Additionally, the renowned tree-based algorithm XGBoost has been employed as an alternative classifier, utilizing features selected through NSC classification. Since XGBoost is not directly supported for the visualizations, we have made use of the flexible `vcr.custom.*` function to prepare the classification results for the visual displays. The required quantities for the classmaps and the MDS Color-Scaled plots have been computed manually, in line with the classifier's operational pattern.

4.3.1 NSC algorithm (pamr)

Fundamentally, the NSC algorithm classifies an unknown sample by evaluating its proximity to a shrunken version of the 'centroid' of known classes. The centroid refers to the multidimensional mean of the feature vectors of samples within a given class.

We enumerate samples as $i = 1.....n$ and variables, which in our case are genes, as $j = 1.....p$. Classes are indicated with $k = 1.....K$. The set of indices of samples that belong to class k is represented by C_k, while n_k indicates the total number of samples in class k. Let x_{ij} represent the expression for the j-th gene of the i-th sample.

The centroid of class k is calculated as $\bar{x}_{kj} = \sum_{i \in C_k} x_{ij}/n_k$, and the overall centroid is $\bar{x_j} = \sum_{i=1}^{n} x_{ij}/n$.

We then define the standardized difference between the class centroid and the overall centroid as follows:

$$\Delta_{kj} = \frac{\bar{x}_{kj} - \bar{x_j}}{m_k \theta_k (s_j + s_0)} \tag{3}$$

Here, s_i denotes the pooled within-class standard deviation (SD) for the i-th gene:

$$s_i^2 = \frac{1}{n-K} \sum_k \sum_{i \in C_k} (x_{ij} - \bar{x}_{kj})^2 \tag{4}$$

The term m_k is set as $\sqrt{1/n_k + 1/n}$ so that $m_k s_i$ equals the estimated SD of $(\bar{x}_{kj} - \bar{x_j})$. The parameter s_0 is a positive offset value, equivalent to the median of the distribution of s_i values, and it serves to penalize genes with expression values near zero. θ_k is an optional scale factor which can take different values for each class, aiming to accommodate possible minor differences in SD between classes. In the standard model, it is set to 1 for all classes, and thus, it is omitted.

By reformulating (2), the class centroid can then be redefined in terms of the overall centroid $\bar{x_i}$ and its deviation Δ_{ik} from it:

$$\bar{x}_{kj} = \bar{x_j} + m_k(s_j + s_0)\Delta_{kj}$$

Each Δ_{ik} is subsequently shrunken toward zero according to the soft thresholding procedure. This technique reduces the magnitude of each Δ_{ik} by a positive amount λ and sets it to 0 if λ greater or equal than it:

$$\Delta'_{kj} = \text{sign}(\Delta_{kj})(|\Delta_{kj}| - \lambda)_+ \tag{5}$$

The symbol $+$ is used to indicate the positive portion of a value ($a_+ = a$ if $a > 0$ and zero otherwise). Now the shrunken class centroid can be represented in terms of the shrunken delta:

$$\bar{x}'_{kj} = \bar{x_j} + m_k(s_j + s_0)\Delta'_{kj} \tag{6}$$

Now the shrinkage toward zero of Δ'_{ik} implies a shrinkage of the class centroid toward the overall centroid. Given a certain threshold value λ the selected genes (also called non-zero delta genes) are those for which at least one shrunken class centroid differs from the overall centroid. The classification decision of the algorithm hinges directly on these variables.

The built model has defined the shrunken centroid $\bar{x}'_{kj}$ for each class based on the training data and the parameter λ. In order to predict the class of a given sample with expression levels

$x_* = (x_{*1}....x_{*p})$ a discriminant score is defined with respect to each class in this way:

$$\delta_k(x_*) = \sum_{j=1}^{p} \frac{(x_{*j} - \bar{x}'_{kj})}{(s_j + s_0)^2} - 2\log \pi_k \tag{7}$$

Where π_k is the prior probability of a sample belonging to a class k, which is typically estimated using the sample priors $\hat{\pi} = n_k/n$. Then the the predicted class $\hat{k}_{x_*}$ is the one attaining the lower discriminant score:

$$\hat{k}_{x_*} = \underset{k \in \{1,...,K\}}{\arg\max}\ \delta_k(x_*)$$

Essentially, the case in question is classified in relation to the Euclidean-nearest shrunken centroid, standardizing the distanceby $(s_j + s_0)$ and incorporating prior probabilities.

Visualizations support for pamr We have designed a specific `vcr.pamr.*` function to prepare results coming from a R pamr classification for the graphical displays.

The R pamr algorithm computes and approximates posterior probabilities directly from the discriminant scores (7), using the softmax function to transform these scores into probabilities, mirroring the approach in Gaussian Discriminant Analysis. For the set of discriminant scores $\delta_g(i)$ calculated for i-th observation, the the set of posterior probabilities is given as:

$$\hat{p}_i(g) = \frac{e^{-\frac{1}{2}\delta_g(i)}}{\sum_{g=1}^{G} e^{-\frac{1}{2}\delta_g(i)}}$$

This makes it possible to calculate the *PAC* and subsequently the *Sil*.

To assess farness for classmaps, the metric $D(i,g)$ which quantifies the distance from observation i to each possible class g from the classifier's perspective should be defined. This measure is unambiguously shaped by the discriminant score that pamr employs to construct its decision rule. Prior probability information is ignored and computation is performed only on the set of

the selected genes J_λ for the chosen shrinkage threshold λ:

$$D(i,g) = \sum_{j \in J_\lambda} \frac{(x_{ij} - \bar{x}'_{gj})}{(s_j + s_0)^2} \text{ with } J_\lambda = \{j \mid \exists g : \bar{x}'_{gj} \neq \bar{x}_j\} \tag{1}$$

For MDS color-scaled plots, pairwise dissimilarities are needed. Just as the distance between an observation and a class is determined, the metric $D(r,s)$ between two generic observations is taken as the Euclidean distance on the features selected, standardizing again by $(s_j + s_0)$:

$$D(r,s) = \sum_{j \in J_\lambda} \frac{(x_{rj} - x_{sj})}{(s_j + s_0)^2} \text{ with } J_\lambda = \{j \mid \exists g : \bar{x}'_{gj} \neq \bar{x}_j\} \tag{2}$$

4.3.2 XGBoost

XGBoost, standing for eXtreme Gradient Boosting, is a powerful, versatile, and highly efficient tree-based ML algorithm. Recently, it has been experiencing substantial recognition in the machine learning domain. Drawing its main principles from the Gradient Boosting framework, it can be employed in both regression and classification predictive modeling tasks.

The algorithm operates by iteratively constructing a collection of weak decision tree models. The final model is a strong predictor that arises from the aggregation of these simpler models. Each successive decision tree is incorporated into the existing ensemble of trees based on a weight assigned by a learning parameter. employs the gradient descent algorithm to minimize a specific loss function, regarding parameters of subsequent decision trees, including splitting variables and split points. The loss function serves as a performance metric for the model, with a lower value indicating superior performance. In the context of multi-class classification, the minimization objective is usually the cross-entropy loss between predicted class probabilities (obtained from the softmax function) and the true labels.

Accessing visualizations for XGBoost There is no specific support available for visualizing classifications made by the XGBoost classifier. To achieve this, we utilize the versatile

`vcr.custom.*` function.

The custom function requires the posterior probabilities for the given classification to generate PAC and Sil values, which are essential for producing silhouette and quasi-residual plots. These posteriors can be automatically generated by the R XGBoost framework. In essence, the comprehensive XGBoost model, when presented with an observation, computes a score for each class that reflects how well the observation fits that class. This is achieved by appropriately weighing and summing the scores given by each tree in the ensemble classifier. These scores, which essentially represent the degree of fit of the observation to each class, are then passed into a specific function - either a sigmoid (in the case of binary classification) or softmax (for multi-class classification). These functions serve to transform the raw scores into a set of proper posterior probabilities that sum to one across the classes.

In order to produce farness values estimates for plotting `classmaps`, the custom function requires a matrix containing distances between each case of the evaluated classification and each possible class (input `DistToclass`). Considering the fundamental nature of the XGBoost classifier as a tree-based classifier, this is done in line with the approach taken by Tibshirani and Raymakers when developing specific support for simple tree-based classifiers and random forests in the R classmap package with vcr.rpart.* and vcr.forest.* (**silhouette2022**). They argue that due to the specific manner in which tree-based classifiers operate and handle data, the selected metric needs to be additive, capable of managing mixed types of variables, and addressing missingness. Furthermore, the metric should account for which variables are predominantly used for splitting and in what order, reflecting the tree's hierarchy, and understand the localized, non-continuous way in which a tree divides the feature space to form its decision boundaries.

In alignment with these considerations, the process of generating dissimilarities with respect to the classes begins by determining pairwise dissimilarities between the set of cases under consideration using the Gower dissimilarity, as implemented by the daisy function in the R

cluster package. For two generic observations i and j, it is defined as:

$$d(i,j) = \frac{\sum_{k=1}^{p} w_k \delta_{ij}^{(k)} d_{ij}^{(k)}}{\sum_{k=1}^{p} w_k \delta_{ij}^{(k)}}$$

Here, the set of weights w_k is determined by the gain values of the fitted XGBoost model. In the context of XGBoost, 'gain' is a metric that quantifies the contribution of each feature to the model. It is calculated based on the improvement in accuracy brought by a feature to the splits or branches for which it is used. Therefore, the gain values represent the proportional influence of each feature on the model, computed as a fraction of the total gain across all splits of that feature. Note that the measure d_{ij} varies if the type of variable changes and δ_{ij} manages missing values. However, in our case of gene expression data, the values are all numerical with no missing values. Given that the Euclidean distance is the reference for numerical variables, this essentially simplifies the metric to a Euclidean weighted dissimilarity. Following this, for each object i and class g under scrutiny, $D(i,g)$ is calculated as the median of the smallest five dissimilarities $d(i,j)$ to all objects j belonging to class g.

Continuing on the topic of farness estimation, when processing a classification on new data - for instance, a test set (handled via vcr.custom.newdata by specifying `newDistToClass`) - the method is slightly modified to only consider the information of a single observation and that from the training data. Initially, all dissimilarities $d(i,h)$ are calculated, where i represents a generic observation from the new data, and h represents any case from the training data. This computation employs the same metric, variable weights and other parameters that were used in the training data. Subsequently, $D(i,g)$ is determined as the median of the smallest k=5 dissimilarities $d(i,h)$ to all objects h that belong to class g in the training data.

4.4 Results

// WORK IN PROGRESS //

5 Conclusion

// WORK IN PROGRESS //

References

Alonso-Betanzos, Amparo et al. (2019). "A Review of Microarray Datasets: Where to Find Them and Specific Characteristics". In: *Microarray Bioinformatics*. Ed. by Verónica Bolón-Canedo and Amparo Alonso-Betanzos. New York, NY: Springer New York, pp. 65–85. ISBN: 978-1-4939-9442-7.

Bruno César Feltes, Joice De Faria Poloni and Márcio Dorn (2021). "Benchmarking and Testing Machine Learning Approaches with BARRA:CuRDa, a Curated RNA-Seq Database for Cancer Research". In: *Journal of Computational Biology* 28, N0. 9, pp. 931–944.

Jakob Raymaekers Peter J. Rousseeuw, Mia Hubert (2022). "Class Maps for Visualizing Classification Results". In: *Technometrics* 64, pp. 151–165.

Jakob Raymaekers, Peter J. Rousseeuw (2022). "Silhouettes and Quasi Residual Plots for Neural Nets and Tree-based Classifiers". In: *Journal of Computational and Graphical Statistics* 31, pp. 1332–1343.

Jakob Raymaekers, Peter Rousseeuw (2023). *classmap*. R package version 1.2.3.

Rousseeuw, Peter J. (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 72, pp. 53–65.

Su Bin Lim Swee Jin Tan, Wan-Teck Lim and Chwee Teck Lim (2018). "A merged lung cancer transcriptome dataset for clinical predictive modeling". In: *Scientific Data* 5.

Tibshirani R. Hastie T., Narasimhan B. and Chu G. (2002). "Diagnosis of multiple cancer types by shrunken centroids of gene expression". In: *Proc. Natl. Acad. Sci. U.S.A* 99, pp. 6567–6572.

– (2003). "Class Prediction by Nearest Shrunken Centroids, with Applications to DNA Microarrays". In: *Statistical Science* 18(1), pp. 104–117.