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 development and effectiveness of Machine Learning algorithms in a specific classification task, as well as into the data structure 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 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 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 powerful tools for extracting meaningful information from complex data structures, finding application across diverse fields such as finance, retail, marketing and biology (Jordan and Mitchell 2015). 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 (Lessmann et al. 2015) to tumor detection in medical imaging (Litjens et al. 2017).

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.

Visual data often offer more intuitive insights than numerical data, as they capitalize on the human brain's capacity to detect patterns and make connections (Tufte 1990). Accordingly, graphical tools can uncover patterns and structures in data that may not be as readily evident through 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 is 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 J. Raymaekers and P. J. Rousseeuw (Raymaekers and P. Rousseeuw 2022; Raymaekers, P. Rousseeuw, and Hubert 2022). These tools, namely "Silhouette plots", "Quasi-residual plots", and "Classmaps", center on the analy-

sis 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-scaled 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 being proposed, which provides adaptive support for accessing visualizations and extends 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 presents a cancer classification problem based on gene expression data, a domain where ML classifiers have increasingly demonstrated their utility. A well-known classifier in the literature, the Nearest Shrunken Centroid (NSC) (Tibshirani et al. 2002) - often referred to as "pamr" in its application to microarray gene-expression data - is employed for this task, specifically on a problem involving the latest 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, shedding light on results for the specific classifiers and offering a comparison between them, thus providing a practical exemplification of their application.

2 The graphical displays for classification

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 (Raymaekers, P. Rousseeuw, and Hubert 2022; Raymaekers and P. Rousseeuw 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 for 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 where the ML algorithm has been trained on, 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 true class of each observation i is known and denoted by g_i.

We assume that once the trained classifier is given an observation i, it can compute the discrete posterior probability $\hat{P}_i(g)$ that describes the probability of the observation i to belong to any class g. This distribution may be computed naturally by the algorithm's design or may be derived coherently based on its decision rule. In either case, the algorithm's class prediction $\hat{g}_i$ for the i-th observation should be the one that maximizes the posterior distribution:

$$\hat{g}_i = \arg\max_{g \in G} \hat{P}_i(g)$$

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 $\tilde{g}_i$:

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

$\tilde{g}_i$ can be interpreted as the best prediction in the eyes of the classifier if the true g_i was not a viable option and is referred to as 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)

Having defined all the necessary quantities, we can now compute 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 both the probability of assigning i to the true class or to the "Best Alternative Class" itself. A meaningful threshold is implicitly defined as a consequence

of (1) and (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 gives the degree of confidence that the considered classifier has in predicting the correct class, in a scale between 0 and 1.

2.2 Silhouette Plot for classification

The silhouette plot was initially proposed in the context of clustering (P. J. 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 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 decisive 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 horizontal 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. On 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.

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

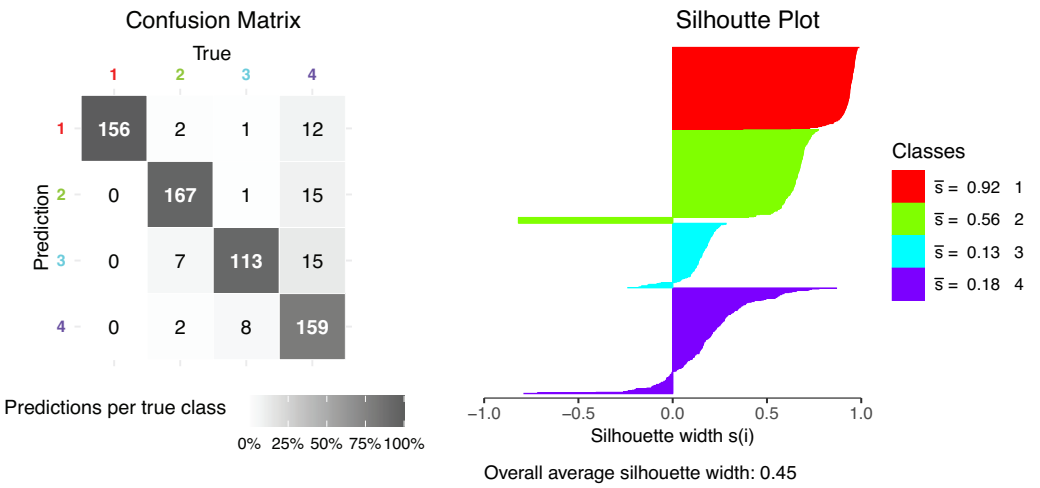

Figure 1: The Confusion Matrix on the left and the Silhoutte Plot on the right. They display the results of the same 4-class classification task on a set where true classes are known.

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, complementing 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 a poor overall prediction quality, 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 its vertical axis and a selected data feature on the horizontal one. Its purpose is to study the association between classification accuracy, as measured by the PAC, and a specific predictor, which may or may not have been part of the model's training. For ease of interpretation, the plot area is divided into two distinct colored sections. Points with PAC values surpassing 0.5, indicating misclassified instances, reside in the white portion of the plot. In contrast, the gray portion includes points with PAC values below 0.5, which represent 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 precision of the classifier. The *PAC*, on average, registers just above 0.2, irrespective of variations in feature 2 values. However, this is not the case for feature 1 where a strong positive correlation with PAC appears. As values in feature 1 increase, the classifier's prediction accuracy tends to deteriorate.

Evaluating these plots can aid to understand and control covariates external to the model, and to decide whether it may be worth incorporating them. Even when certain information cannot be integrated, it is beneficial to identify a covariate that could potentially challenge the model's accuracy. These details are critical for a mindful model deployment. 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.

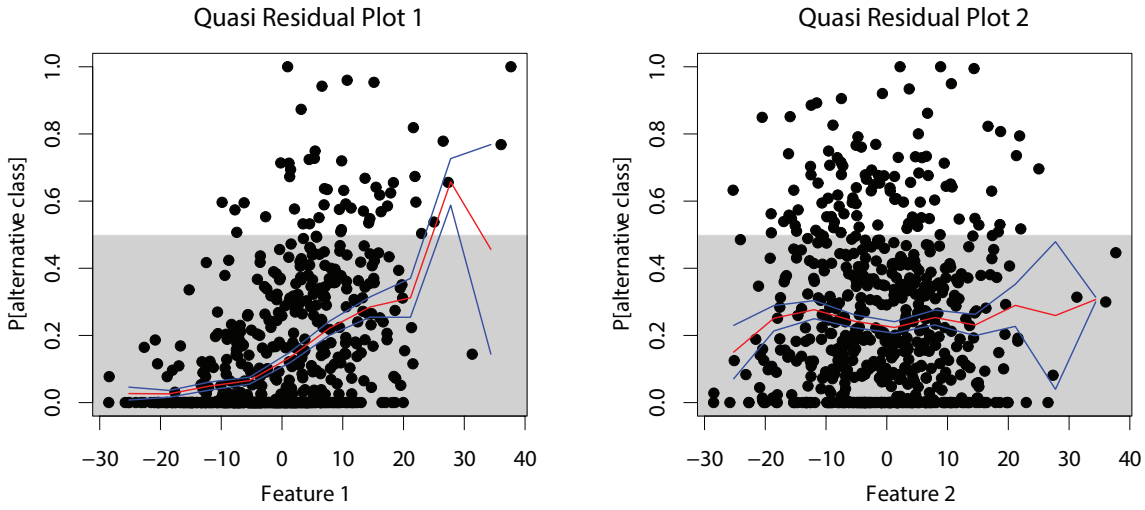

Figure 2: Two examples of quasi-residual plots with average PAC values in red and average plus/minus standard error in blue.

It is important to note, given that this plot does not distinguish between classes, that the emergence of a significant correlation between a certain feature and the PAC does not 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 specific class, which the classifier predicts either better or worse than average. Lastly, the lack of correlation between PAC and a feature could not imply that the feature is irrelevant to the model.

2.4 Classmap Plot

A classmap is a specific type of quasi-residual plot that is class-specific and displays observations within the considered class according to the known true labeling. The horizontal 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 has learned by 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 $D(i, g_i)$ indicating the distance for the observation i from its true class g_i. Subsequently, $D(x, g_i)$ is introduced. This quantity represents the distance of a randomly generated observation x, 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 "Silhouttes and Quasi Residual Plots for Neural Nets and Tree-based Classifiers" (Raymaekers and P. Rousseeuw 2022). 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 is 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 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 consistent 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 is plausible for the dataset under review.

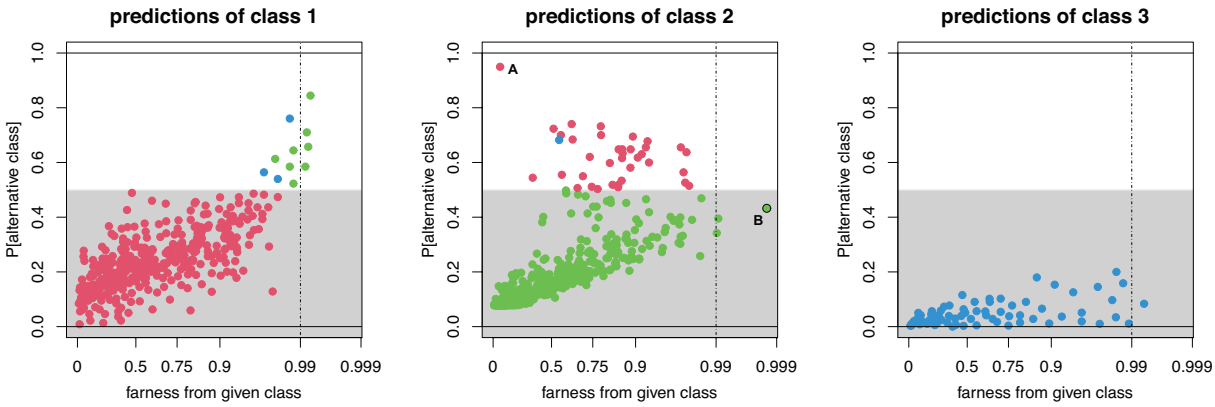

Figure 3: Classmaps on the results of a 3-class classification problem

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 The novel 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.

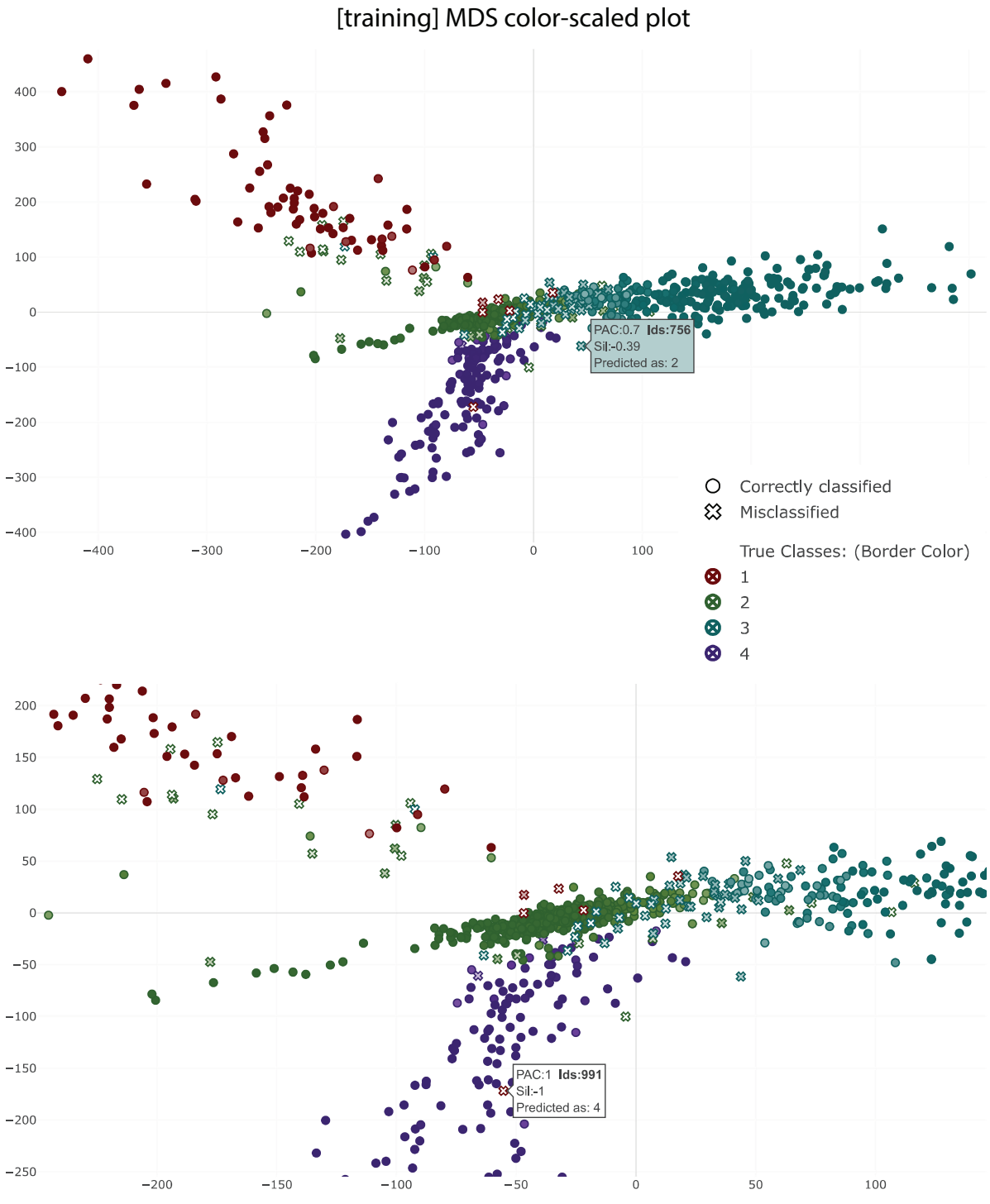

Figure 4: The MDS-color scaled plot (above) displays the results of a four-class classification problem on the training set, with a zoomed-in view (below).

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 $D(i, j)$ to be devised, with i and j 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.

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 an observation unveils its IDs, denoting its location in the data matrix, the specific PAC/SIL values it scores, and its class prediction.

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 Accessing the visualizations in R

3.1 "classmap" package

The functions need to produce the three reviewed graphical displays, namely the silhouette, quasi-residual plot, and classmap, are provided in the R package `classmap` (Raymaekers and P. Rousseeuw 2021), released on CRAN, where Raymaekers and Rousseeuw have meticulously implemented their proposals. The corresponding functions names are `silplot`, `qresplot`, and `classmap`. These are not immediately ready-to-use with general results of a given classification. This is due to the processing required, tied to the employed ML algorithm, to obtain the required quantities for each classified case, specifically the *PAC* and the $farness$ value.

To compute the *PAC*, it is necessary to collect 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 ML algorithms, including Discriminant Analysis, Random Forest, K-Nearest Neighbors, 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.*.*`, where the second asterisk stands for either `train` or `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 algorithms. 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 from the eponymous package, and `vcr.svm.*` for support vector machines requires the fit of the specific `svm` function in the `e1071` package.

This rigidity does not offer an immediate method for researchers to use the plotting functions within the `classmap` package framework with algorithms of their 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 train-

ing 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.

3.2 "classmapExt" package

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.*` whose 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, the package 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 evaluated in section 4, where the development of the specific `vcr.pamr.*` function is also explained, as well as a case use of `vcr.custom.*`.

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 ML 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, the discussed visualizations can provide valuable insights and help decipher complex nuances within the data.

Moreover, in this context, ML classifiers often serve a dual purpose. Beyond building a mere predictive model, they are employed to enrich our comprehension of the underlying data and its inherent structure, including any substructures present. Graphical displays may contribute meaningfully to this objective by shedding light on the intricacies of the data and elucidating the relationships and trends therein, not only aiding accurate classification but also augmenting 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 (Feltes, Poloni, and Dorn 2021).

After raw data are collected following a selected experiment design, they undergo a series

of pre-processing 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 pre-processed, the dataset can be fed into proper 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 differ from the training data.

- **Dimensionality and feature selection**: The high-dimensionality of genetic data, which are 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 sta-

bility 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 not only to build accurate models but also to 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**: Non-biological variations may be introduced during data collection stages in different batches. When merging datasets, these effects can cause discrepancies, complicating data interpretation and analysis. Hence, careful handling and rigorous normalization methods are essential when dealing with such situations. (Lim et al. 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 (Alharbi and Vakanski 2023).

There are several advantages in 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

We have made the decision to utilize a dataset that leverages RNA-Seq, a NGS technique, to determine the gene-expression profiles of the samples. The adoption of RNA-Seq technology is becoming increasingly prevalent in gene expression quantification due to the comprehensive insights it provides into the transcriptome (Whang, Gerstein, and Snyder 2009). Furthermore, recent RNA-Seq datasets include a greater number of observations and a wealth of valuable additional covariates about the individuals considered in the studies, which can be very useful.

Unlike microarray-based techniques, which rely on predetermined probes associated to known genes, RNA-Seq employs a sequencing approach that enables a broader, more detailed examination of gene expression. This approach not only quantifies absolute gene expression levels but also facilitates the identification of novel transcripts and alternative splice variants.

Despite fundamental differences in the technologies and data collection methods utilized in RNA-Seq and microarray approaches, the end products of both techniques can be formatted compatibly. From a collection of RNA-Seq experiments, it is feasible to generate matrices that quantify gene expression levels across a predetermined set of genes, typically encompassing the entire genome. This is akin to the gene expression matrices derived from a series of microarray experiments. This common format facilitates the integration of RNA-Seq data, once pre-processed, into existing analytical pipelines designed for microarray data.

Both methods require an essential tailored normalization step in the pre-processing stage. This step serves to adjust potential technical disparities between samples, guaranteeing a neutral platform for ensuing comparative analyses. Furthermore, this normalization facilitates the transition from raw measurements to comparable gene expression metrics.

The specific dataset selected originates from The Cancer Genome Atlas (TCGA) project data, accessible via the NCI's Genomic Data Commons (GDC) repository. Our study focuses on lung

cancer, selecting lung tissue samples from the two lung cancer types available in the project: Lung Squamous Cell Carcinoma (LUSC) and Lung Adenocarcinoma (LUAD). Additionally, the small portion of available normal lung tissue samples (NORM) is included. Lung cancer is the leading cause of cancer-related deaths worldwide, making it a significant focus of study. The two subtypes, LUSC and LUAD, cover about 85% of the total cases and are often challenging to differentiate due to their overlapping clinical presentations (Herbst, Morgensztern, and Boshoff 2018).

We assembled a gene expression matrix utilizing Transcripts Per Million (TPM) normalized gene counts, which outputs from the standard GDC mRNA RNA-Seq pre-processing pipeline (see Figure 5) for each individual sample. TPM counts are included in the open-access end-of-pre-processing 'TSV' file available for each sample, along with raw feature counts, Fragments Per Kilobase of transcript per Million mapped reads (FPKM), and upper quartile normalized FPKM (FPKM-UQ) counts. This comprehensive file provides an array of normalized and raw count data, enabling downstream nalysis and processing.

The most recent data, specifically from Data Release 37 published on 29th March 2023, were downloaded. The related GDC pre-processing pipeline begins with the raw FASTQ files, typically two due to paired-end sequencing, sourced from the RNA-Seq experiment. These files undergo a two-pass STAR alignment, with quality assessments conducted using FASTQC for pre-alignment and Picard Tools for post-alignment evaluations. The most recent assembly of the human genome, GRCh38, is employed as reference for read alignment. To identify genes and other genomic features, it is utilized the GENCODE annotation file, version 36. The unstranded and stranded counts-per-genes results produced by the STAR software are retained. TPM normalized counts are calculated using unstranded raw counts, accounting for both sequencing depth and gene length: making differences in the experiment library size irrelavant and ensuring a between-sample comparable metric. In a given sample, the TPM value for a

Figure 5: GDC mRNA RNA-Seq pre-processing pipeline that leads to gene-expression quantification, as per Data Release 37.

specific gene, denoted as f, within a defined set of genes F, is computed using the formula:

$$TPM_f = \frac{(C_f \times 1e^3 / L_f)}{\sum_{f \in F}(C_f \times 1e^3 / L_f)} \times 1e^6$$

Here, C_f represents the raw count of unstranded reads aligning to the gene f, and L_f stands for the cumulative length of exons for that gene. The scaling factor $1e^3$ is applied to convert read counts into reads per kilobase (RPK), and the scaling factor $1e^6$ is used to normalize the sum of all TPM values in the sample to one million.

The final gene expression matrix produced comprises 1153 observations, with TPM values that span 19962 protein-coding genes. The annotation sourced from GDC presents a more comprehensive set of about 60,660 gene features. However, we have opted to focus our attention on protein-coding genes to secure robust and biologically pertinent results from the classification models. The dataset consists of 541 LUAD, 502 LUSC, and 110 NORM instances.

The feature matrix is complemented by an extensive set of covariates that provide detailed characterizations of the individuals from whom the samples were procured. These covariates stem from two primary sources: the GDC repository and the TCGA Pan-Cancer Atlas project (The Cancer Genome Atlas Research Network et al. 2013). The data from the Pan-Cancer project, accessed via cBioPortal, provide a wealth of valuable metrics, reflecting the whole-genome profile of the individuals under study. Examples of such measures include the count of mutations, the total fraction of the genome altered, and the MSI MANTIS score.

The GDC processing pipeline also produces auxillary statistics related to the alignment of sequencing reads against the reference genome in its final quantification output. These are:

- **N. unmapped**: number of reads that were not mapped to the reference genome. There are several reasons why a read might fail to map. For instance, the read sequence might be of poor quality, it may contain an excess of errors, or it could originate from a region not covered by the reference genome, such as a virus or other foreign contaminant.

- **N. multimapped** number of reads that were mapped to multiple locations on the reference genome. This can occur in regions of the genome with high similarity or sequence repeats.

- **N. noFeature** number of reads that, despite being successfully mapped to the reference genome, did not overlap with any annotated genomic feature, such as a gene, exon, or intron. These reads could potentially originate from intergenic regions or yet-to-be annotated genes.

- **N. ambiguos** number of reads that overlapped with multiple genomic features, making it unclear to which feature the read should be assigned. For instance, if a read overlaps with two overlapping genes on opposite strands, it would be deemed ambiguous.

These statistics have been combined into a unified metric that we term the "alignment Quality Index" (*AQI*). This single quantitative measure serves as an indicator of the quality and effectiveness of the alignment process. Furthermore, given the use of an accurate reference

genome assembly and a comprehensive annotation, and considering that alignment is a final downstream step in RNA-seq pre-processing, the *AQI* can also indirectly reflect the overall quality of the sample and the sequencing process. It is computed as follows:

$$\text{AQI} = \frac{\frac{1}{2} \cdot \text{unmapped} + \frac{1}{6} \cdot (\text{multimapped} + \text{noFeature} + \text{ambigous})}{\text{Total number of mapped reads}}$$

A greater weight is attributed to unmapped reads, which are especially troublesome since they do not align to any part of the reference genome and thus provide no actionable genomic information. This typically signals a likely presence of low-quality or contaminated reads. Although it is possible these reads might represent sequences absent in the reference genome, it is less probable given that alignment can handle a certain level of sequence variation due to mutations and variants.

A higher *AQI* value indicates a larger proportion of problematic reads to map, highlighting a sub-optimal and less effective alignment and hinting potential issues with the sample. Conversely, lower *AQI* suggests a stronger and more successfull alignment.

This type of metric can provide valuable quality-related insights further down the RNA-seq pre-processing pipeline. This is especially useful when only the final quantification output and some mapping stage statistics are available, as in our situation, and when other indicators like base quality, sequencing depth, and facets of experimental design are not directly obtainable.

The curated introduced dataset consists of the latest gene-expression RNA-Seq data and a wide array of covariates from one of the largest cancer studies available. Further enriched by features from the pan-cancer study and read alignment quality metrics, it offers a comprehensive and up-to-date source for our gene expression classification task.

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 et al. 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 et al. 2003). Here we have decided to repurpose this method, testing its prowess on RNA-seq data. The NSC algorithm offers the advantage of natively including 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 `vcr.pamr.*` functions to specifically support `pamr` classifications in R, taking into consideration the classifier's specific mechanisms. Plots are drawn for classifications results on both the train and the test set, into which the dataset was preventively split according to a 80/20 rule.

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 "classmap" plots and the "MDS Color-scaled" plots have been manually computed 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 produced fitting the training data. 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 $f = 1.....p$. Classes are indicated with $g = 1.....G$. The set of indices of samples that belong to class g is represented by C_g, while n_g indicates the total number of samples in class g. Let x_{if} represent the expression for the f-th gene of the i-th sample.

The centroid of class g is calculated as $\bar{x}_{gf} = \sum_{i \in C_g} x_{if}/n_g$, and the overall centroid is $\bar{x}_f = \sum_{i=1}^{n} x_{if}/n$.

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

$$\Delta_{gf} = \frac{\bar{x}_{gf} - \bar{x}_f}{m_g \theta_g (s_f + s_0)} \tag{3}$$

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

$$s_f^2 = \frac{1}{n-G} \sum_g \sum_{i \in C_g} (x_{if} - \bar{x}_{gf})^2 \tag{4}$$

The term m_g is set as $\sqrt{1/n_g + 1/n}$ so that $m_g s_f$ equals the estimated SD of $(\bar{x}_{gf} - \bar{x_f})$. The parameter s_0 is a positive offset value, equivalent to the median of the distribution of s_f values, and it serves to penalize genes with expression values near zero. θ_g 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 (3), the class centroid can then be defined in terms of the overall centroid $\bar{x_f}$ and its deviation Δ_{gf} from it:

$$\bar{x}_{gf} = \bar{x}_f + m_k (s_f + s_0) \Delta_{gf}$$

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

greater or equal than it:

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

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 defined substituting the delta with its shrunken version:

$$\bar{x}'_{gf} = \bar{x}_f + m_g(s_f + s_0)\Delta'_{gf} \tag{6}$$

The shrinkage toward zero of Δ'_{gf} 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 optimal value for λ is usually determined by evaluating the classifier accuracy in cross-validation.

The built model has defined the shrunken centroid $\bar{x}'_{gf}$ 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 as follows:

$$\delta_g(x_*) = \sum_{f=1}^{p} \frac{(x_{*f} - \bar{x}'_{gf})}{(s_f + s_0)^2} - 2\log \pi_g \tag{7}$$

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

$$\hat{g}_{x_*} = \underset{g \in \{1,...,G\}}{\arg\max}\ \delta_g(x_*)$$

Essentially, the case in question is classified in relation to the Euclidean-nearest shrunken centroid, standardizing the distance by $(s_f + s_0)$ 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 in its decision rule. Prior probability information is ignored and computation is performed only on the set of the selected genes F_λ for the chosen shrinkage threshold λ:

$$D(i,g) = \sum_{f \in F_\lambda} \frac{(x_{if} - \bar{x}'_{gf})^2}{(s_f + s_0)^2} \text{ with } F_\lambda = \{f \mid \exists g : \bar{x}'_{gf} \neq \bar{x}_f\} \tag{1}$$

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

$$D(i,j) = \sum_{f \in F_\lambda} \frac{(x_{if} - x_{jf})^2}{(s_f + s_0)^2} \text{ with } F_\lambda = \{f \mid \exists g : \bar{x}'_{gf} \neq \bar{x}_f\} \tag{2}$$

4.3.2 XGBoost

XGBoost, standing for eXtreme Gradient Boosting, is a powerful, versatile, and highly efficient tree-based ML algorithm (Chen and Guestrin 2016). 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. It employs the gradient descent algorithm to minimize a specific loss function, regarding parameters of the subsequent decision tree, 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.*` (Raymaekers and P. Rousseeuw 2022). 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 (Gower 1971), as implemented by the `daisy` function in the R `cluster` package. For any two given observations, i and j, over a set of variables that spans from $f = 1....p$, it is defined as:

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

Here, the set of weights w_f 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 final 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 $d_{ij}^{(f)}$ could change in relation to what type of variable f is, and $\delta_{ij}^{(f)}$ 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 dissimilarity to an Euclidean weighted distance. Following this, for each object i and class g under scrutiny, $D(i,g)$ is calculated as the median of the smallest $k = 5$ 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 for 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.

Ultimately, to draw the MDS color-scale plot, the matrix of pairwise dissimilarities has been already thoughtfully-computed in the 'farness computation process. The only difference is that, this time, when dealing with new test data the pairwise dissimilarities are computed just taking into consideration the whole test set.

4.4 Results

NSC. Figure 6 shows the misclassification error rate, as estimated through a 10-fold cross-validation procedure, for a set of 30 different λ threshold parameter values. Both the overall and within-class error are displayed. As the value of λ increases, the model exerts a stronger shrinkage towards the overall centroid, resulting in a diminished count of selected genes. We singled out two λ values for in-depth analysis of the corresponding models. One is the threshold value of 17.38, which achieves the minimum overall misclassification rate (6.4%) and is

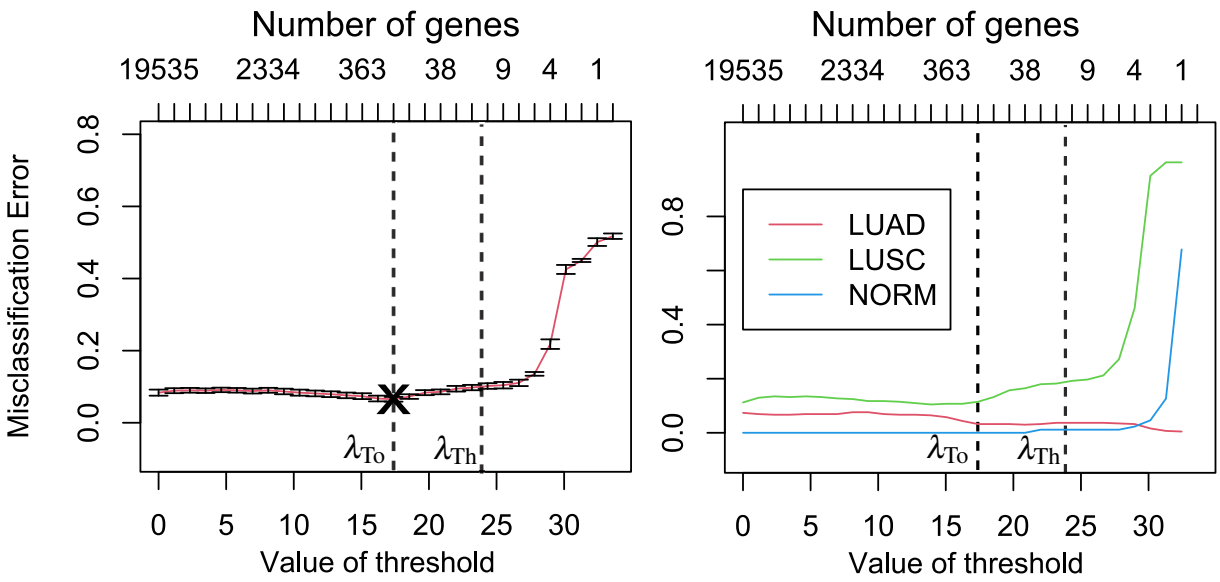

Figure 6: Misclassification error rates determined by a 10-fold cross validation performed on the training set for different threshold values. The two vertical dashed lines represents the values selected.

referred to as λ_{To}. With this amount of shrinkage 150 genes are identified for sample discrimination. The other is a higher threshold value of 24.33, which maintains a relatively low overall error rate (10.1%), despite the fact that it selects only 14 genes. This other value for the parameter is referred to as λ_{Th}. Performance was optimized in both models by utilizing ($\theta_{LUAD} = 2.09, \theta_{LUSC} = 1, \theta_{NORM} = 2.09$) as a scale factor (adaptive thresholding), in lieu of the unit vector.

Figure 7 comprises a panel featuring silhouette, classmap, and MDS color-scaled plots, illustrating the classifications performed on both the training and testing sets by the two models with λ_{To} and λ_{Th}. A glance at the silhouette plots reveals consistent outcomes between the training and testing sets within both models, a result that aligns with the robust-to-overfitting design of the algorithm. While the LUAD class registers fewer misclassified instances, a noticeable proportion of errors are recorded for the LUSC samples, with a slight increase in the test set. No instances of NORM are misclassified in the optimal model, and only one in the sub-optimal one.

For the λ_{To} model, instances are either classified with a high degree of confidence into the cor-

rect class, or erroneously predicted to a wrong class without much ambiguity. This observation contrasts with the λ_{Th} model, which only shows high silhouette values for NORM instances. Indeed, silhouette widths for correctly classified LUAD instances hover around 0.5, and there is a general decrease in silhouette values for correct LUSC predictions. Misclassified instances within this class do not appear as drastically off-target as those in the threshold-optimal model, which delivers more dichotomous results.

The additional shrinkage does not seem to substantially worsen the apparent accuracy of the predictions, but it does affect their quality, as measured by the *PAC*, to a greater extent. The average value of the silhouette width decreases from 0.77 to 0.43.

Shifting our attention to the classmap plots, the misclassification of LUSC samples for LUAD samples becomes conspicuously clear, especially in the sub-optimal model. This misclassification appears to be the ending result of an increasing pattern in PAC values as instances drift away from the squamous cell cancer gene expression profile built prototype. Such a general trend of increase of *PAC* as 'farness' increases is a typical trend we would expect for this kind of plot. Intriguingly, in both models, the strongly misclassified LUAD instances are extremely close to their correct class in the decision space. Nevertheless, these instances are even closer to the NORM class and are thus categorized as such, indicating a critical proximity of the two classes with solid decision boundaries separating them - an aspect further highlighted by the newly presented plot in this paper. In fact, upon examing the MDS color-scaled plots, the relationships delineated between the classes by the trained classifier become clearer. NORM and LUSC classes stand out as distinctly separated entities, with LUAD class seemingly positioned in between them. The optimal model draws sharp decision boundaries among classes, yet when considering the higher threshold, a more gradual, blended separation is seen between LUAD and LUSC. Instead, the margin between NORM and LUAD holds its rigidity. Ultimately, the differentiation between LUSC and LUAD seems to pose a tangible challenge for the NSC algorithm on this dataset.

Using quasi-residual plots, we carried out an in-depth examination of the association that covariates—derived from the extensive auxiliary data for each individual—have with the *PAC* measured accuracy of the predictions. We directed our attention towards studying this aspect for the Th model, considering that for the optimal model, the range of different *PAC*/*Sil* values is limited. Such binary levels of confidence render these types of plots less meaningful. Figure 8 presents six such plots for the training set, portraying *PAC* against selected covariates. One of these covariates, "histological type", is qualitative and was numerically coded prior to generating the plot. For such categorical covariates, while we can observe differences in *PAC* across levels, discerning trends is not feasible due to the lack of ordinality in the feature. We did not identify any notable associations between the covariates and *PAC*, thereby suggesting that the disease-discrimination capacity, as derived from the gene expression profile, seems to be independent of covariates such as gender, sex, mutation count, smoking habits, morphology, among others. However, a positive correlation was discerned between the constructed *AQI* and *PAC*. To delve deeper into this correlation, we also examined quasi-residual plots within class (Figure 9), revealing that higher *AQI* scores particularly challenge the predictions for LUSC instances. This may imply that, considering *AQI* as a valid indicator of overall sequencing quality, the 14-genes NSC begins to struggle to distinguish LUSC from LUAD instances, supplementing the insights from our previous considerations.

Additionally, *PAC* was linearly-regressed against the expression levels of all selected and non-selected genes, not identifying any relevant correlation.

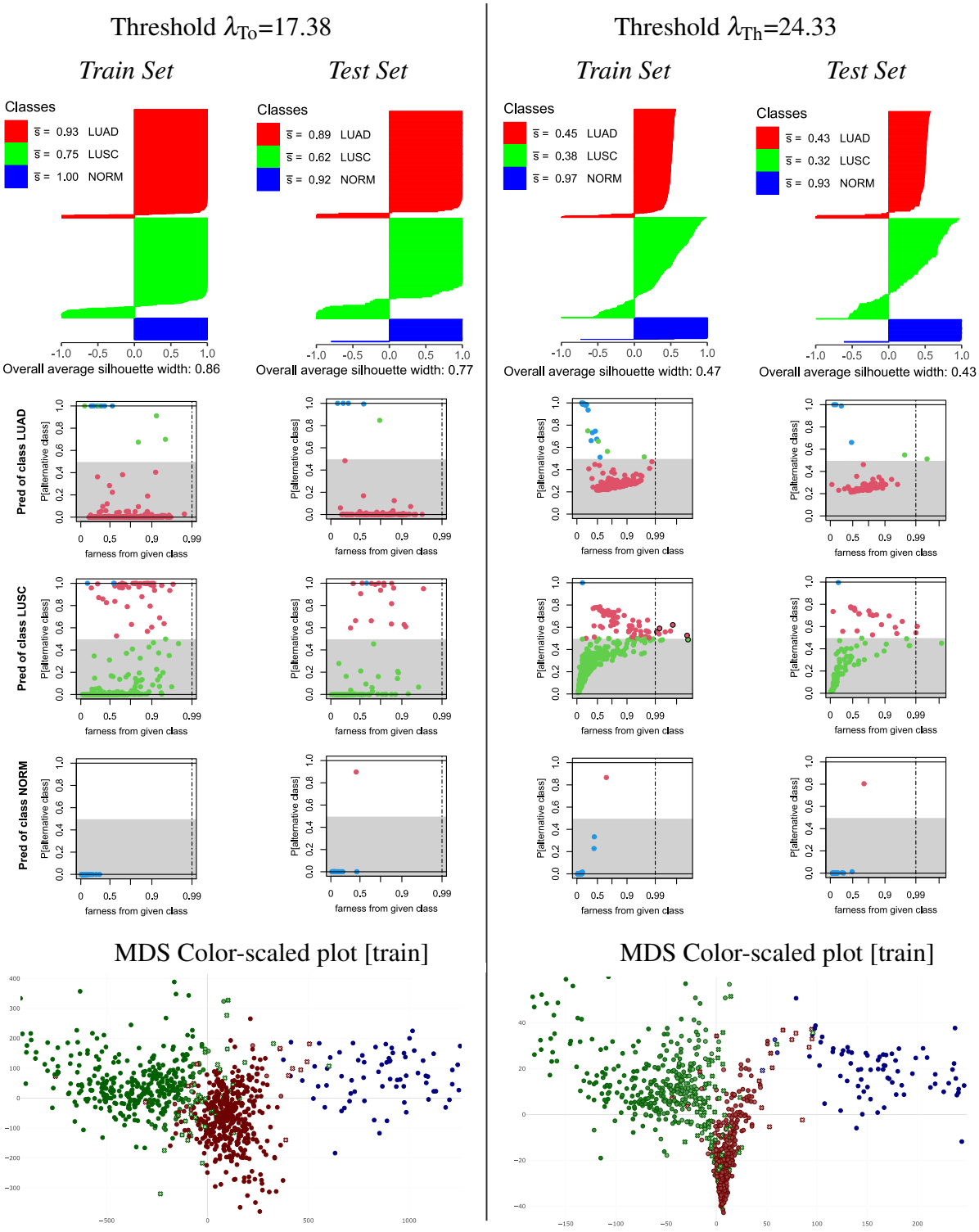

Figure 7: A comparative panel of Silhouettes, Classmaps, and MDS-Colorscale plots for classifications performed by two NSC classifiers, each set with a different threshold.

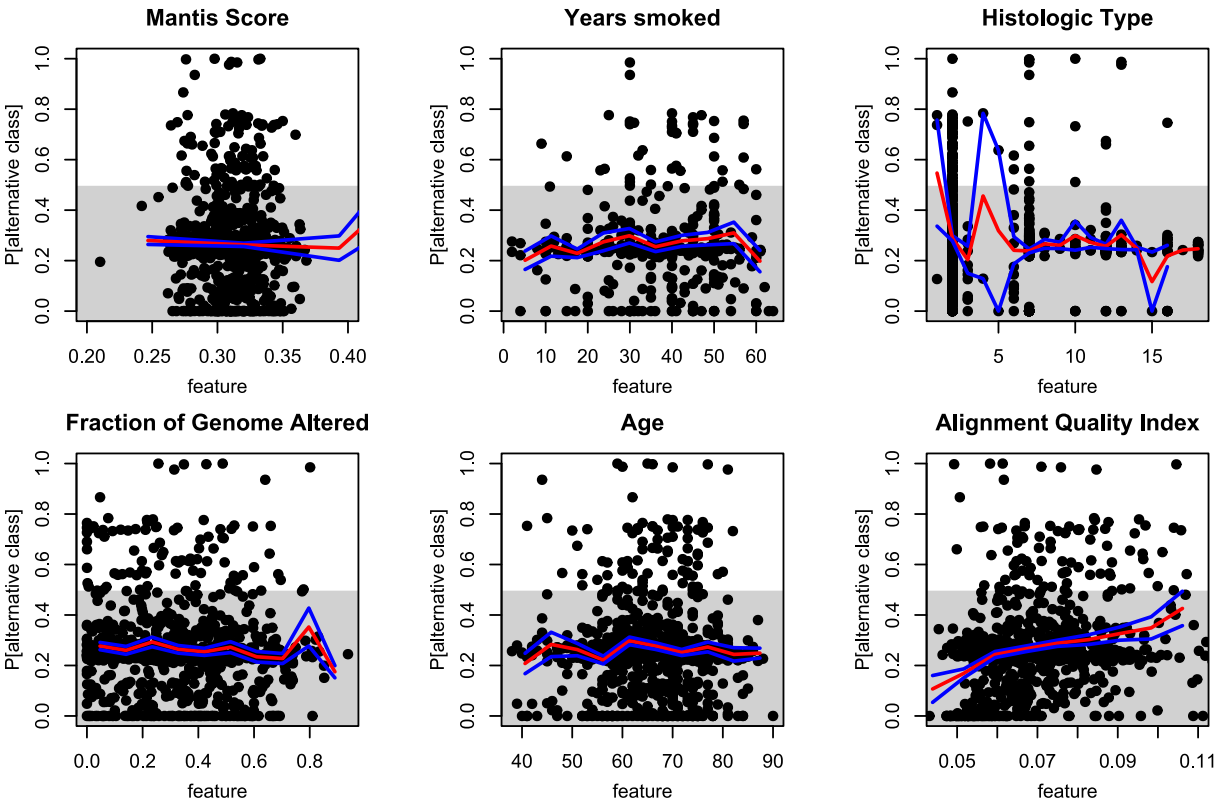

Figure 8: A selection of Quasi -residual plots for the classification on the training set by the λ_{Th} model

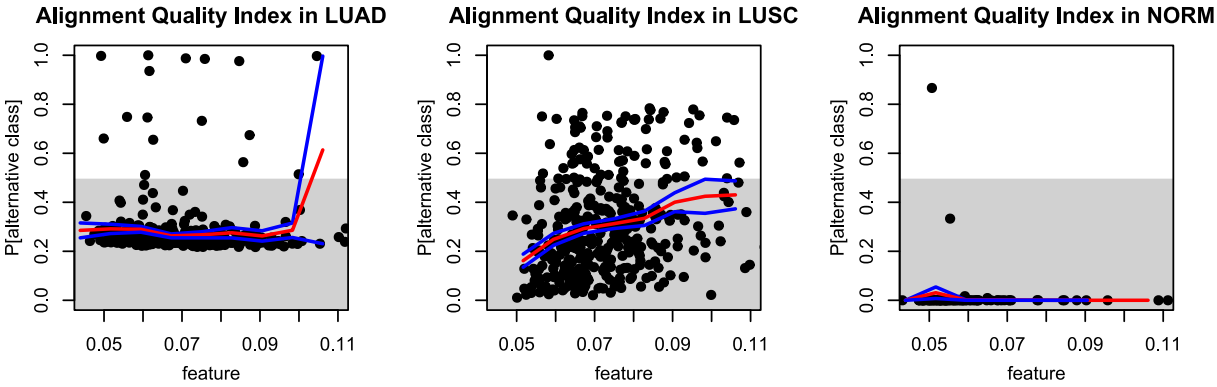

Figure 9: Within-class Quasi-residual plots for the classification on the training set by the λ_{Th} model with respect to the Alignment Quality Index.

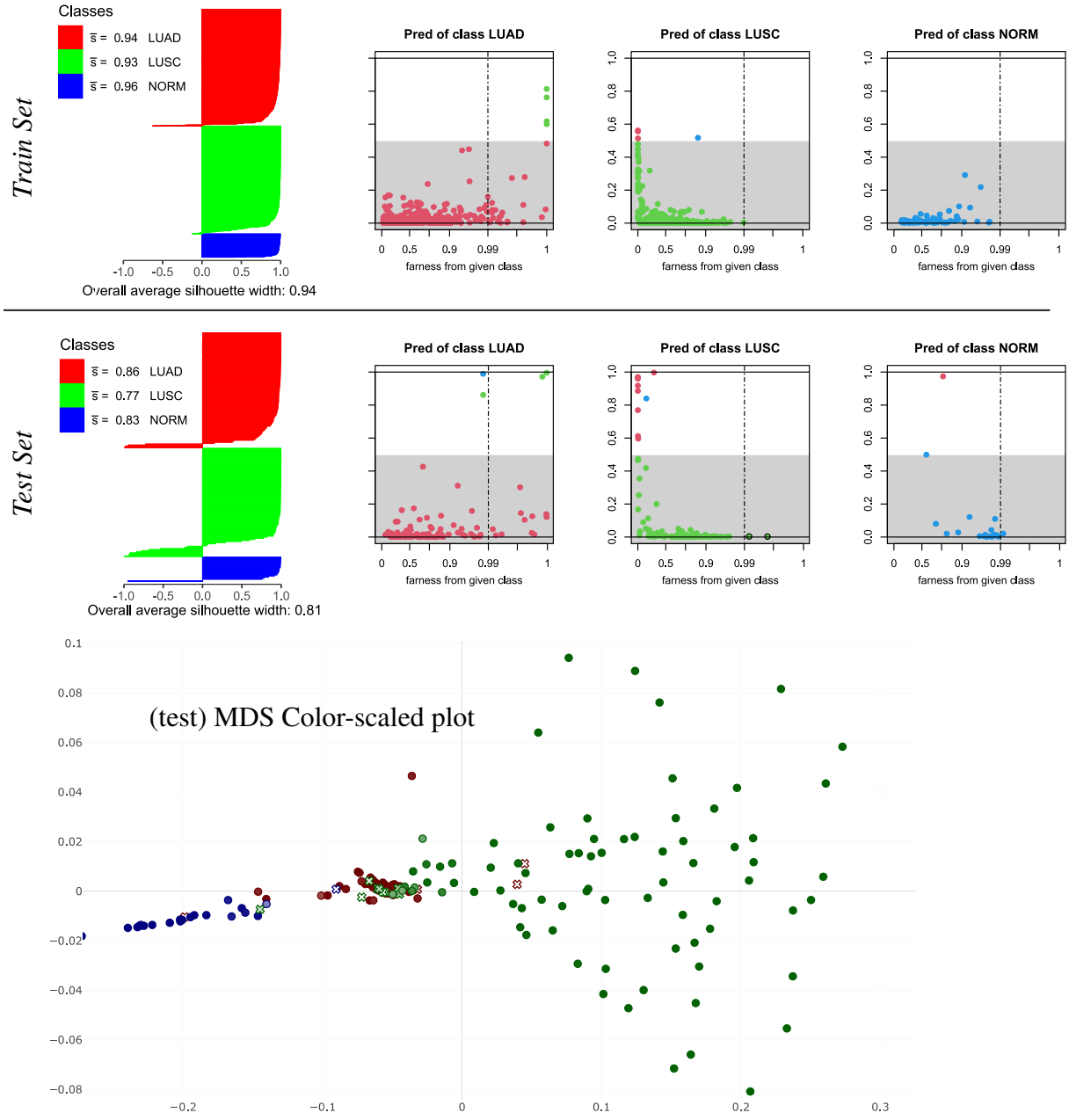

Figure 10: A comparative panel of Silhouettes, Classmaps, and MDS-Colorscale plots for classifications on both Train and Test set performed by the XGBoost classifier.

XGBoost. The XGBoost classifier was trained on the 14 genes identified by NSC at the λ_{Th} threshold. Additionally, the *AQI* index was included as a training feature due to its potential impact on predictions showed for the NSC model, with the hope that XGBoost could leverage it to enhance efficiency.

Algorithm hyperparameters were selected based on cross-validation outcomes on the training

set. Figure 10 reports the graphical tools applied to the classifications of the available observations by the optimally-tuned XGBoost model. The classifier excels in the training set, misclassifying only 8 out of the 922 total instances and achieving a near-perfect overall silhouette score of 0.94. On the test set the performance worsen but it still remains far better of the λ_{Th} NSC model, recording a misclassification error rate of 6.5% and a overall silhoutte score 0.81. These results are even better than those of the best λ_{To} NSC model that employs 10 times the number of genes. Silhouette width values are again rather dichotomous here, either very high, close to 1, or very low, close to -1 for misclassifications.

Classmaps for the test set shows "healthier" LUAD predictions, with only a handful of instances far from the class center being misclassified. Conversely, LUSC class suffers from misclassifications pertaining instances that lie close to the built LUSC prototype. This scenario recalls the situation between NORM and LUAD classes for the NSC models, characterized by close class positioning in space with distinct, strong decision boundaries, but now occurring between LUSC and LUAD samples. Although NORM class instances are largely classified correctly, they mostly lie rather far from the classifier's ideal class representation. However, this does not pose an issue as they are still classified with correct strong conviction. In the training set, NORM instances are more nearly-located to the relative class prototype, which is to be expected as the model was fitted to this samples.

The MDS color-scaled plot for the test set maintains a similar spatial positioning of the classes as NSC, with LUSC and NORM further apart. This time, however, the instances of NORM and LUAD are better discriminated. The boundaries between LUSC and LUAD continue to pose challenges. Interestingly, the LUSC class appears more dispersed in space and less closely clustered than the other two classes within the discriminative subspace. Two LUSC outliers, identified on the classmaps (black contour), are accurately classified despite their considerable distance, as they are located in the direction opposite to the boundaries with the LUAD class.

Figure 11 presents a heatmap of the expression values for the 14 selected genes and the AQI in-

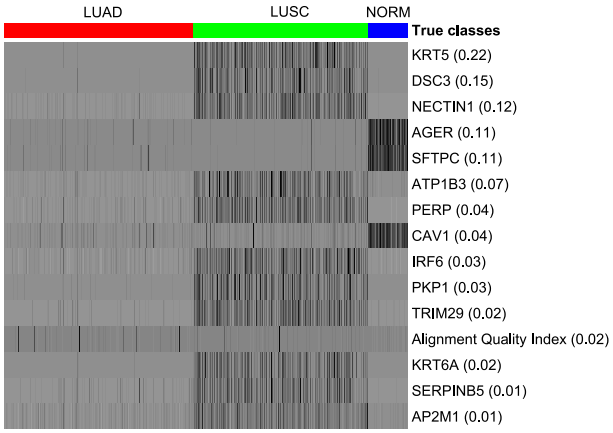

Figure 11: Heatmap of the 14 genes selected by the λ_{Th} NSC model, along with the AQI index, across the entire TCGA-LUNG dataset. Between the parenthesis are reported the "gain" values of the corresponding features for the XGBoost model that has been trained on them.

dex values, arranged by true classes for the entire dataset of 1153 observations. Vertically, genes are ordered by their fractional contribution to the final XGBoost model, as per 'gain' values, indicated in parentheses, obtained directly from `xgb.importance` output. These values have been employed to compute pairwise dissimilarities among observations for `mdscolorscale`, aimed at defining a space as close as possible to the classifier's decision space and as the basis for the computation of 'farness' for `classmap`. The genes AGER, SFTPC, and CAV1 display significant differential expression between normal and cancerous lung tissue samples, with cancerous samples generally attaining lower expression values (less intense color representation). KRT5, scoring a gain value of 0.22, emerges as the most influential gene for the model. Its expression, along with DSC3, NECTIN1, ATP1b3, and PERP—key contributors to model performance—enables effective discrimination between LUSC and LUAD samples. However, these genes exhibit a similar expression profile in LUAD and NORM tissues, thereby providing little contribution to further discrimination, which could account for the difficulties faced by the NSC model in differentiating between them. The *AQI* does not display noticeable systematic differences across classes, except for a few outliers in LUAD with particularly high values. The algorithm, as indicated by the 'gain' score of 0.02, also seems to use it sparingly.

5 Conclusion

We have provided a comprehensive overview of recently proposed graphical tools for classification problems. Silplots, with their distinctive intuitiveness, provide a qualitative dimension to prediction accuracy, serving as a insightful complement to ubiquitous confusion matrices. Classmaps, along with the newly introduced MDS Color-scaled plots, depict that accuracy within the spatial perception that the trained classifier has on the cases, decoding the relationships it has built among classes in its decision space. Quasi-residual plots delve deeper into the algorithm's strengths and limitations by exploring the association between *PAC* and features of the data. The application of these tools is clarified further by their use in a current, challenging example. Users are offered flexible access to the R framework that enables them to draw such displays to enhance the understanding of their data and the ML classifier they employed.

6 Supplementary materials

`classmapExt` R package, with the extensions proposed in the paper, is available at `https://github.com/LLazzar/classmapExt`. The dataset and analysis scripts pertaining to the gene-expression cancer classification example can be found at `https://github.com/LLazzar/LUNG-TCGA-Classifications-Graphical-Analysis`.

References

Alharbi, F. and A. Vakanski (2023). "Machine Learning Methods for Cancer Classification Using Gene Expression Data: A Review". In: *Bioengineering* 10.2.

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

Chen, T. and C. Guestrin (2016). “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794.

Feltes, B. C., J. D. F. Poloni, and M. Dorn (2021). “Benchmarking and Testing Machine Learning Approaches with BARRA:CuRDa, a Curated RNA-Seq Database for Cancer Research”. In: *Journal of Computational Biology* 28.9, pp. 931–944.

Gower, J. C. (1971). “A General Coefficient of Similarity and Some of Its Properties”. In: *Biometrics* 27.4, pp. 857–871.

Herbst, R. S., D. Morgensztern, and C. Boshoff (2018). “The biology and management of non-small cell lung cancer”. In: *Nature* 553.7689, pp. 446–454.

Jordan, M. I. and T. M. Mitchell (2015). “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349, pp. 255–260.

Lessmann, S. et al. (2015). “Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research”. In: *European Journal of Operational Research* 247, pp. 124–136.

Lim, S. B. et al. (2018). “A merged lung cancer transcriptome dataset for clinical predictive modeling”. In: *Scientific Data* 5.

Litjens, G. et al. (2017). “A survey on deep learning in medical image analysis”. In: *Medical Image Analysis* 42, pp. 60–88.

Raymaekers, J. and P. Rousseeuw (2021). *classmap*. R package version 1.2.3.

– (2022). “Silhouettes and Quasi Residual Plots for Neural Nets and Tree-based Classifiers”. In: *Journal of Computational and Graphical Statistics* 31, pp. 1332–1343.

Raymaekers, J., P. Rousseeuw, and M. Hubert (2022). “Class Maps for Visualizing Classification Results”. In: *Technometrics* 64, pp. 151–165.

Rousseeuw, P. 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.

The Cancer Genome Atlas Research Network and, et al. (2013). “The Cancer Genome Atlas Pan-Cancer analysis project”. In: *Nature Genetics* 45, pp. 1113–1120.

Tibshirani, R. et al. (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.

Tufte, E. R. (1990). *Envisioning Information*. Graphics Press.

Whang, Z., M. Gerstein, and M. Snyder (2009). “RNA-Seq: a revolutionary tool for transcriptomics”. In: *Nature Reviews Genetics* 10, pp. 57–63.