reval: a Python package to determine the best number of clusters with stability-based relative clustering validation

Isotta Landi*[1], Veronica Mandelli[1,2], and Michael V. Lombardo[1,3]

[1] Laboratory for Autism and Neurodevelopmental Disorders, Center for Neuroscience and Cognitive Systems @UniTn, Istituto Italiano di Tecnologia, Rovereto, Italy.

[2] Center for Mind/Brain Sciences, University of Trento, Rovereto, Italy.

[3] Autism Research Centre, Department of Psychiatry, University of Cambridge, Cambridge, United Kingdom.

Abstract

Determining the number of clusters that best partitions a dataset can be a challenging task because of 1) the lack of a priori information within an unsupervised learning framework; and 2) the absence of a unique clustering validation approach to evaluate clustering solutions. Here we present **reval**: a Python package that leverages stability-based relative clustering validation methods to determine best clustering solutions.

Statistical software, both in R and Python, usually rely on internal validation metrics, such as the *silhouette index*, to select the number of clusters that best fits the data. Meanwhile, open-source software solutions that easily implement relative clustering techniques are lacking. Internal validation methods exploit characteristics of the data itself to produce a result, whereas relative approaches attempt to leverage the unknown underlying distribution of data points looking for a replicable and generalizable clustering solution.

The implementation of relative validation solutions can further the theory of clustering by enriching the already available methods that can be used to investigate clustering results in different situations and for different data distributions. This work aims at contributing to this effort by developing a stability-based method that selects the best clustering solution as the one that replicates, via supervised learning, on unseen subsets of data. The package works with multiple clustering and classification algorithms, hence allowing further assessment of the stability of different clustering mechanisms.

Keywords— stability-based relative validation, clustering, unsupervised learning

*Correspondence: landi.isotta@gmail.com, @IsottaLandi

Introduction

Clustering algorithms identify intrinsic subgroups in a dataset by arranging together elements that show smaller pairwise dissimilarities relative to other subgroups [1]. They are one of a number of machine learning methods that do unsupervised learning with the aim of identifying patterns in the data in the absence of supervised/external knowledge. The process of determining the number of clusters in a dataset is often challenging because of the lack of a priori information and of a unique clustering validation approach to evaluate clustering solutions. Attempts to address this challenge leverage both relative and internal cluster validity methods. In particular, relative criteria aim to compare clustering solutions obtained with different parameter settings, whereas internal criteria focus on quantities and features inherent to a grouping solution [2]. While a variety of software packages contain internal cluster validity methods and metrics, open-source software solutions to easily implement relative clustering techniques are lacking. Here we present the `reval` Python package (pronounced "reh-val", like the word "revel"), that implements stability-based validation of clustering solutions to determine the number of clusters that best partitions the data, as described by Lange and colleagues [3].

Several methods exist to determine the number of clusters based on internal criteria. For example, the elbow method [4] selects the number of clusters for which the within-cluster variability decrement is minimum. Another popular method using internal criteria is the silhouette-based approach [5]. This method maximizes cluster cohesion and separation - that is, how similar an object is to other elements of the same cluster compared to elements of other clusters. Silhouette is one of many other statistics that have been developed using internal criteria to suggest what the optimal number of clusters is. As an example, the **NbClust** R library [6] compiles 30 different internal metrics and users have the option to compute all or a subset of these metrics and use a majority vote rule to select the optimal number of clusters. In contrast to internal criteria, relative validation techniques transform the problem of selecting the best number of clusters into a model selection problem. In particular, stability-based methods return the number of clusters that minimizes the expected distance between clustering solutions obtained for different datasets. Several methods are available [7] to 1) generate the datasets, e.g., random subsampling of the original dataset [8], or adding random noise [9]; 2) compare clustering solutions, e.g., overlapping subsamples [8]; and 3) compute clustering distances, e.g., the consensus index by Vinh and Epps [10].

The method proposed by Lange et al. [3] has the advantage of transforming the unsupervised setting into a classification problem and guiding selection through the minimization of prediction error. In this way, the returned solution can also be tested on an unseen test set to inform replicability. First, a dataset is split into training and validation sets and then independently partitioned into clusters. Second, training set

labels are used within supervised classification methods to learn how to best predict the labels. Applying the classification model to the validation set, the model's predicted labels are then compared to the actual clustering labels derived from the validation set. This procedure is repeated using cross-validation and the optimal number of clusters corresponds to the number of clusters that minimizes the prediction error. Prediction performance can be defined in different ways. One straightforward choice is the equivalent of the 0-1 loss in supervised classification [3, 7], namely, the misclassification error. Nevertheless, other choices are possible, for example, Tibshirani and colleagues used prediction strength - that is, the proportion of observation pairs in the validation set that are assigned to the same cluster by both the clustering algorithm and the partition based on the training set centers [11].

Internal and relative indices can exhibit similar behavior with respect to clustering errors, with the advantage of the former being less computationally expensive [12]. However, in the case of complex models and clusters, an approach based on minimization of prediction error is more indicative of clustering performance, because internal indices tend to fail to correlate well with errors [12]. Moreover, in the case of the model selection approach within a cross-validation framework of Lange and colleagues [3] another advantage is that it allows for selection of the solution that best generalizes to new unseen subsets of data in a supervised fashion and for investigation of replicability of the clustering results in an automated fashion. On the contrary, internal indices are highly tied to the characteristics of the data at hand and thus can run the risk of overfitting. Hence, they can be not well suited when the aim is to assess the underlying structure of a dataset.

Libraries and methods for the automated selection of the best number of clusters are available in both Python and R. The **yellowbrick** Python visual analysis and diagnostic tool suite [13] includes the class `KElbowVisualizer()` that implements the elbow method to determine the best number of clusters. In R, we have already introduced **NbClust** [6] which relies on internal validation indices. Moreover, based on relative validation there are **clValid** [14] and **cstab** [15] that apply stability-based relative validation models. The former works especially with highly correlated high-throughput genomic data and computes stability measures comparing clustering solutions based on full data and data with a single column removed. Moreover, it also enables validation with internal measures. The latter implements the selection of the best number of clusters via model-based and model-free clustering instability [16] using a bootstrap approach.

The **reval** package contributes to this landscape by implementing a stability-based approach that can be easily applied to different datasets using multiple clustering and classification algorithms. The evaluation of the best model on unseen dataset is also easily applicable and classification accuracy is returned.

Models and software

Stability measure

The notion of stability given by Lange et al. [3] is used to assess solutions of clustering algorithms based on the rationale that true clusters are those that can always be identified by a clustering algorithm when applied to different datasets from the same generating process. Formally, let $\mathbf{X} = (X_1, \ldots, X_n) \in \mathcal{X}^n$ be a dataset, which we want to partition into k clusters with a clustering algorithm $\mathcal{A}_k$. We write:

$$\mathcal{A}_k(\mathbf{X}) = \mathbf{Y} \tag{1}$$

with $\mathbf{Y} = (Y_1, \ldots, Y_n)$ the clustering solution such that $Y_i = j$, $j \in \{1, \ldots, k\}$, if sample i is assigned to cluster j by algorithm $\mathcal{A}_k$.

We can now introduce ϕ as a classifier that can be trained on labeled data and predict labels for a new, unseen dataset. If we consider $(\mathbf{X}, \mathbf{Y})$ as the training set, then ϕ can be applied to a new set $\mathbf{X}' = (X'_1, \ldots, X'_n)$ and produce labels $\phi(\mathbf{X}') = (\phi(X'_i))_{i=1,\ldots,n}$. If we now apply the clustering algorithm to $\mathbf{X}'$, we obtain a clustering solution $\mathcal{A}_k(\mathbf{X}') = \mathbf{Y}'$ that can be directly compared to the solution $\phi(\mathbf{X}')$ obtained from the classifier. We define:

$$d_{S_k}(\phi(\mathbf{X}'), \mathbf{Y}') = \min_{\sigma \in S_k} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{\sigma(\phi(X'_i)) \neq Y'_i\}} \tag{2}$$

with S_k the set of all possible permutations of k elements, as the dissimilarity between the two solutions. Supervised labels are permuted to overcome the non-uniqueness of clustering labeling and σ is the permutation that minimizes the solutions dissimilarity.

Averaging out the distance between any pair of partitions $\mathbf{X}$, $\mathbf{X}'$ from Eq. 2 we obtain the stability index of the clustering algorithm:

$$\mathcal{S}(\mathcal{A}_k) = \mathsf{E}_{\mathbf{X},\mathbf{X}'}[d_{S_k}(\phi(\mathbf{X}'), \mathbf{Y}')] \tag{3}$$

The stability index $\mathcal{S}(\mathcal{A}_k)$ ranges in $[0, 1]$, with lower values indicating stable and reproducible solutions [3].

We now observe that the similarity measure in Eq. 3 scales with k. If we were to obtain a misclassification rate of 0.5 for $k = 2$, the performance would be the same as a random predictor. On the contrary, the same misclassification rate for $k > 2$ would indicate a better performance than random guessing. To overcome this, we define a random labeling algorithm $\mathcal{R}_k$, which assigns an object to cluster j with probability $\frac{1}{k}$ and we normalize the stability index with the one obtained from random labeling algorithms, i.e., $\mathcal{S}(\mathcal{R}_k)$. The

normalized stability index is defined as:

$$\bar{\mathcal{S}}(\mathcal{A}_k) = \frac{\mathcal{S}(\mathcal{A}_k)}{\mathcal{S}(\mathcal{R}_k)} \tag{4}$$

In conclusion, the best number of clusters k^* is the one that minimizes the normalized stability $\bar{\mathcal{S}}$. In order to obtain k^* we need to: 1) iteratively split data into $\mathbf{X}$, $\mathbf{X}'$ to estimate stability; and 2) estimate $\bar{\mathcal{S}}$ over a range of different values of k.

To avoid overfitting and measure the data distribution's inherent stability, the classifier ϕ should be selected as the ϕ^* that minimizes the misclassification error, i.e.,

$$\phi^* = \text{argmin}_{\phi}\, \mathsf{E}_{\mathbf{X},\mathbf{X}'}\Big[\min_{\sigma \in S_k} \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{\sigma(\phi(X'_i)) \neq Y'_i\}}\Big] \tag{5}$$

Intuitively, the classifier should mimic the clustering algorithm as much as possible to minimize the mismatch between clustering and classification algorithms [3].

Algorithm description

In the following, we present the pseudo-code that illustrates the algorithm implemented in **reval**, which allows the user to: 1) automatically select the best number of clusters for a dataset $\mathbf{X}$ via repeated cross-validation (see Algorithm 1); 2) compute the classification accuracy obtained generalizing the solution to an held-out dataset (see Algorithm 2). An overview of the framework is reported in Figure 1.

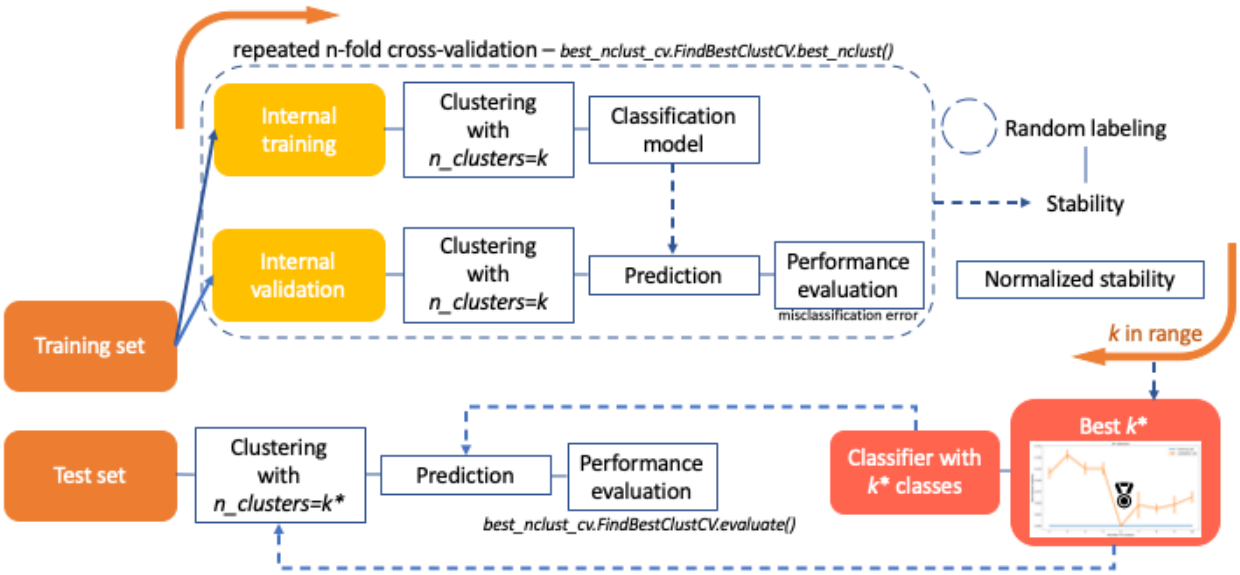


Figure 1: **reval** implementation overview. Repeated cross-validation procedure is included within the dashed circle and it is repeated for different numbers of clusters k as indicated by the orange arrows (Algorithm 1). The clustering algorithm with the best number of clusters k^* selected is evaluated on a held-out dataset (Algorithm 2).

As preparation, dataset $\mathbf{X}$ needs to be split into training $\mathbf{X}_{tr}$ and test $\mathbf{X}_{ts}$ sets. Furthermore, clustering $\mathcal{A}$ and classification ϕ algorithms should be selected. Let n be the number of folds for cross-validation, N the number of repetitions, and k the number of clusters. In Algorithm 1 we indicate with $\mathbf{X}_{itr}^{i_j}$ and $\mathbf{X}_{val}^{i_j}$ the internal training and validation splits of training set $\mathbf{X}_{tr}$, respectively, for cross-validation ith fold split at the jth shuffled repetition. $\mathcal{R}_k$ refers to the random labeling algorithm, as before.

Algorithm 1: Select best number of clusters.

Input: $\mathbf{X}_{tr}$, $\mathcal{A}$, ϕ, K, n, N, *random_iter*

Result: k^*

for $k = 1, \ldots, K$ **do**

  for $i = 1, \ldots, n$ **do**

    for $j = 1, \ldots, N$ **do**

      Find clustering solution $\mathcal{A}_k(\mathbf{X}_{itr}^{i_j}) = \mathbf{Y}_{itr}^{i_j}$ and train ϕ on $(\mathbf{X}_{itr}^{i_j}, \mathbf{Y}_{itr}^{i_j})$;

      Compute $\phi_{i_j}(\mathbf{X}_{val}^{i_j})$ and $\mathcal{A}_k(\mathbf{X}_{val}^{i_j}) = \mathbf{Y}_{val}^{i_j}$;

      Select permutation $\bar{\sigma}_{i_j} \in S_k$ that yields to minimum dissimilarity $d_{\bar{\sigma}_{i_j}}(\phi_{i_j}(\mathbf{X}_{val}^{i_j}), \mathbf{Y}_{val}^{i_j})$;

      for $r = 1, \ldots,$ *random_iter* **do**

        Train ϕ on $(\mathbf{X}_{itr}^{i_j}, \mathcal{R}_k(\mathbf{Y}_{itr}^{i_j}))$;

        Compute $d_{\bar{\sigma}_{r_j}}(\phi_{r_j}(\mathbf{X}_{val}^{i_j}), \mathbf{Y}_{val}^{i_j}))$ as before;

      end

      Compute $d_{i_j} = d_{\bar{\sigma}_{i_j}} / \operatorname{Avg}_{r=1}^{\text{random\_iter}}(d_{\bar{\sigma}_{r_j}})$;

    end

  end

  Compute normalized stability $\bar{\mathcal{S}}_k = \operatorname{Avg}_{j=1}^{N} \operatorname{Avg}_{i=1}^{n} d_{i_j}$;

end

Return k^* s.t. $\min_k \bar{\mathcal{S}}_k$.

Algorithm 2: Test best solution on unseen data.

Input: $\mathbf{X}_{tr}$, $\mathbf{X}_{ts}$, k^*, $\mathcal{A}_{k*}$, ϕ

Result: classification accuracy

Find clustering solution $\mathcal{A}_{k^*}(\mathbf{X}_{tr}) = \mathbf{Y}_{tr}$ and train ϕ on $(\mathbf{X}_{tr}, \mathbf{Y}_{tr})$;

Compute $\phi_{k^*}(\mathbf{X}_{ts})$ and $\mathcal{A}_{k^*}(\mathbf{X}_{ts}) = \mathbf{Y}_{ts}$;

Compute the accuracy with permuted clustering labels for consistency between training and test sets:

$$\text{ACC} = \max_{\sigma \in S_{k^*}} \text{Avg}_{i=1}^{n\_samples} \mathbb{1}_{\{\phi((X_{ts})_i) = \sigma((Y_{ts})_i)\}}$$

Return ACC $\in [0, 1]$.

Software implementation

The **reval** package has three modules:

- `relative_validation`: This module includes training and test methods that return misclassification errors (Eq. 3) obtained by comparing classification labels and clustering labels. Within this module, the `RelativeValidation.rndlabel_traineval()` method implements random labeling that allows users to compute the asymptotic misclassification rate (Eq. 4). As suggested by Lange and colleagues [3], we used the Kuhn-Munkres algorithm [17, 18] to return the permutation that minimizes the misclassification error. However, differently from the work of Lange and colleagues [3], **reval** permutes the clustering labels instead of the classification labels, i.e., the solution dissimilarity in Eq. 2 becomes:

$$d_{S_k}(\phi(\mathbf{X}'), \mathbf{Y}') = \min_{\sigma \in S_k} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{\phi(X'_i) \neq \sigma(Y'_i)\}}$$

  This approach allows the test set to preserve the partition structure of the training set, to better investigate the result's replicability and to aid in visual comparison. The `kuhn_munkres_algorithm()` function can be found in *utils* file.

- `best_nclust_cv`: This module implements replicable repeated cross-validation procedure (if `iter_cv` parameter of `FindBestClustCV.best_nclust()` method is greater than 1, classic cross-validation otherwise) and returns the best number of clusters along with the normalized stability scores, obtained from the average of the misclassification scores divided by the asymptotic misclassification rate. Repeated cross-validation leads to unbiased stability estimates and it can also be performed stratifying the repeated randomized splits according to a desired variable. The `FindBestClustCV.evaluate()` method applies the fitted model with the returned number of clusters to the held-out dataset.

- `visualization`: This module includes the `plot_metrics()` function to plot cross-validated performance metrics with 95% confidence intervals for a varying number of clustering solutions.

A more thorough description of the code and its usage can be find at `https://reval.readthedocs.io/en/latest/code_description.html`. **reval** works with the **scikit-learn** Python library for machine learning [19]. In particular, among clustering methods, users can select those with the `n_clusters` parameter, i.e., `KMeans()`, `SpectralClustering()`, and `AgglomerativeClustering()` from `sklearn.cluster`. Moreover, any classifier from **scikit-learn** can be selected (e.g., `KNeighborsClassifier()` from `sklearn.neighbors`).

Simulations

To investigate **reval** performance, we first present an ensemble learning procedure similar to that by Rodriguez et al. [20], where we consider 18 different datasets from the UCI Machine Learning repository [21] and apply all possible classifier-clustering combinations and report the best results (i.e., those with minimum misclassification error) along with the number of clusters selected. This was done to investigate the best combinations of clustering and classification algorithms, to find the best partitions on state-of-the-art datasets, and to add considerations or caveats about the method. Furthermore, we report **reval** performance on two other examples. To show how to use **reval**, we give an example applied to simulated non-overlapping Gaussian blobs. To showcase how **reval** performs at detecting the true number of clusters in a more complex and real-world dataset, we show how it is applied to the hand-written digits dataset.

Ensemble learning

We considered 18 datasets from the UCI Machine Learning repository [21]. In Table 1 we report the dataset names along with the number of samples, features, and inherent classes. We applied the relative validation algorithm with different combinations of classifier-clustering algorithms. In particular, for clustering we selected: hierarchical clustering (HC) with Ward's method and Euclidean distance; k-means clustering with Euclidean distance; spectral clustering with radial basis function kernel and k-means to assign labels in the embedding space. Among classifiers we opted for: k-nearest neighbors (KNN) with 1 neighbor and Euclidean distance; random forest (RF) classifier with 100 estimators; support vector machines (SVM) with $C = 1$ and $\gamma = \frac{1}{\text{N samples}}$; and logistic regression (LogReg). To improve the performance, in addition to raw datasets, we repeatedly run the experiments after preprocessing with: 1) standard scaler, which removes the mean and scales to unit variance; 2) uniform manifold approximation and projection (UMAP) algorithm [22] for dimensionality reduction. The parameters are chosen according to those suggested in the documentation (see

Dataset	Samples	Features	Classes
banknote	1372	4	2
biodegradation	1055	41	2
breast cancer (WI)	683	9	2
climate	540	18	2
ecoli	336	7	8
forest type	523	27	4
glass	214	9	6
ionosphere	351	34	2
iris	150	4	3
leaf	340	14	30
liver disorders	345	5	2
movement	360	90	15
parkinsons	195	22	2
seeds	210	7	3
transfusion	748	4	2
urban land cover	675	147	9
wholesale	440	7	3
yeast	1484	8	10

Table 1: Benchmark datasets from the UCI machine learning repository.

`https://umap-learn.readthedocs.io/en/latest/clustering.html`); and 3) UMAP and standard scaler. We ran the algorithm with one replication of 5-fold cross validation and 100 random labeling iterations. The number of clusters ranges from 2 to $n_C + 2$, where n_C is the number of classes for each dataset.

Table 2 reports the best solutions selected as those reporting minimum stability along with the correct number of clusters. If no experiment identified the correct number of classes, we chose, among the solutions with minimum stability for each preprocessed dataset, the one with maximum adjusted mutual information (AMI), which measures the similarity of two labeling of the same data, irrespective of label order. AMI ranges in $[0, 1]$, with perfect match when equal to 1.

We observe that the correct number of classes was identified in the 77% of cases ($N = 14$). Moreover, 11 out of the 14 results reporting the correct number of clusters returned k-means clustering, and either SVM, RF, or KNN as classifiers. Of these, only 3 utilized directly the raw data with no preprocessing, while the rest required either scaled or UMAP-preprocessed datasets. Because k-means works well with center-based spherical clusters and usually cannot find a good representation if clusters are very elongated or have complicated shapes, data preprocessing can relax this issue. From this experiment, it emerged that UMAP can be used as a preprocessing tool in this sense, in that it unwraps manifolds to find manifold boundaries. Nevertheless, because each dataset has its own intrinsic characteristics, preprocessing steps must be chosen with care to avoid breaking clusters into several erroneous small spherical clusters (see example with k-means at `https://umap-learn.readthedocs.io/en/latest/clustering.html`). It has been observed [7] that if the number of clusters k is too large with respect to the true clusters, the k-means algorithm tends to be

Dataset	Classes	Performance				
		Best model	Stability (CI)	AMI	Clusters	Preprocessing
banknote	2	KMeans + KNN/LogReg/SVM/RF	0.0	0.75	2	UMAP + scaling
biodegradation	2	Spectral + SVM	0.37 (0.44)	0.13	2	UMAP + scaling
breast cancer (WI)	2	KMeans + KNN	0.03 (0.03)	0.76	2	scaling
climate	2	KMeans + RF	0.16 (0.08)	−0.005	2	UMAP
ecoli	8	**KMeans + KNN**	0.27 (0.27)	0.70	4	scaling
forest type	4	KMeans + SVM	0.29 (0.24)	0.55	4	raw
glass	6	KMeans + RF	0.21 (0.15)	0.23	6	UMAP + scaling
ionosphere	2	KMeans + SVM	0.02 (0.03)	0.21	2	raw
iris	3	**all combinations**	0.0	0.73	2	UMAP + scaling
leaf	30	**KMeans + KNN**	0.0	0.22	2	UMAP + scaling
liver disorders	2	KMeans + KNN	0.39 (0.32)	0.04	2	scaling
movement	15	**HC + SVM**	0.31 (0.15)	0.43	7	scaling
parkinsons	2	KMeans + SVM	0.35 (0.30)	0.23	2	scaling
seeds	3	KMeans + SVM	0.07 (0.08)	0.79	3	scaling
transfusion	2	KMeans + KNN	0.28 (0.15)	0.02	2	raw
urban land cover	9	Spectral + RF	0.99 (0.006)	−0.001	9	scaling
wholesale	3	KMeans + RF	0.05 (0.07)	0.002	3	UMAP
yeast	10	HC + RF	0.23 (0.10)	0.24	10	UMAP + scaling

Table 2: Ensemble learning **reval** results on benchmark datasets from the UCI machine learning repository. In bold results from the experiments that failed in identifying the correct number of clusters. HC = hierarchical clustering; KNN = k-nearest neighbors; RF = random forest; SVM = support vector machine; UMAP = uniform manifold approximation and projection

.

unstable. On the contrary, if k is smaller or equal to the true number of clusters, the algorithm tends to be stable. Therefore, it is argued that k-means stability depends on the number of true clusters in the dataset, which should be on the order of 10 to provide stable solutions. This seems to hold when we look at the UCI datasets with ≥ 10 classes. It is clear how the returned number of clusters providing stability it is either correct or too small. In conclusion, ensemble learning is easily and effectively implemented with **reval** and can be handy to avoid a priori choosing of a classifier. Rather, the choice of a classifier that best mimics the clustering should be one that minimizes the expected stability [3].

Blobs

To illustrate how a user can utilize **reval**, we generate 1000 samples with 2 features, divided into 5 classes with `make_blobs()` function from `sklearn.datasets`. This function generates isotropic Gaussian blobs for clustering (see Figure 2).

```
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from reval.best_nclust_cv import FindBestClustCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from sklearn.metrics import zero_one_loss, adjusted_mutual_info_score
```

```
from reval.visualization import plot_metrics
import matplotlib.pyplot as plt
from reval.utils import kuhn_munkres_algorithm

data = make_blobs(1000, 2, 5, center_box=(-20, 20),
                  random_state=42)
plt.figure(figsize=(6, 4))
plt.scatter(data[0][:, 0],
            data[0][:, 1],
            c=data[1], cmap='rainbow_r')
plt.title("Blobs dataset (N=1000)")
plt.show()
```

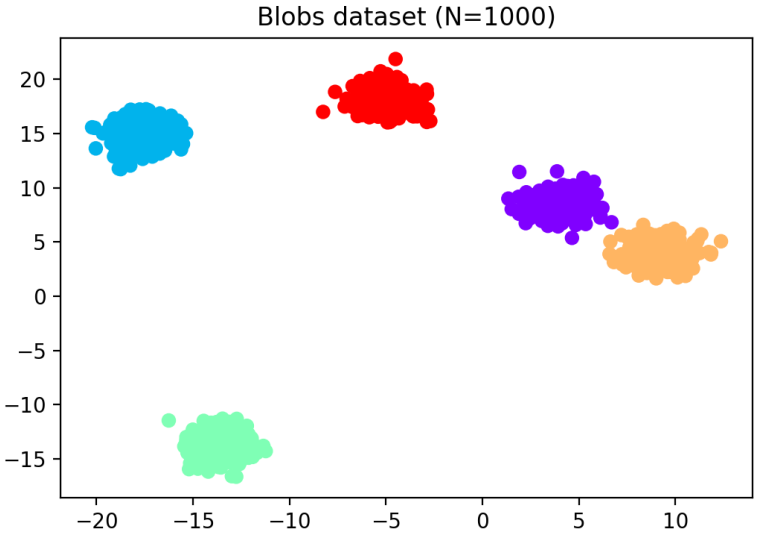


Figure 2: 1000 samples of 5 isotropic Gaussian blobs with 2 features. Uniform manifold approximation and projection (UMAP) is used for visualization purposes.

The dataset is first split into training and test sets (30% of data).

```
X_tr, X_ts, y_tr, y_ts = train_test_split(data[0],
                                          data[1],
                                          test_size=0.30,
                                          random_state=42,
                                          stratify=data[1])
```

Given the considerations from the Ensemble learning section, we choose k-means with Euclidean distance as

clustering method and KNN as classifier. We initialize `FindBestClustCV()` class with the clustering method and classifier selected, number of cross-validation folds equal to 10, number of clusters ranging from 2 to 7, and number of random labeling iterations equal to 100. We then call `best_nclust()` function to select the best number of clusters with 10 repetitions of cross-validation, and `evaluate()` to test the model on the test set.

```
classifier = KNeighborsClassifier(n_neighbors=5)
clustering = KMeans()


findbestclust = FindBestClustCV(nfold=10,
                                nclust_range=[2, 7],
                                s=classifier,
                                c=clustering,
                                nrand=100)
metrics, nbest, _ = findbestclust.best_nclust(X_tr, iter_cv=10,
                                              strat_vect=y_tr)
out = findbestclust.evaluate(X_tr, X_ts, nbest)
```

Now we permute the test set output labels to match true labels in order to compute the classification accuracy. Moreover, we compute the adjusted mutual information (AMI) score between true and clustering labels. We also report the best number of clusters returned by the algorithm, the prediction accuracy obtained by the model on the test set comparing predicted and clustering labels, and the validation set normalized stability (with mean and error).

```
perm_lab = kuhn_munkres_algorithm(y_ts, out.test_cllab)


print(f"Best number of clusters: {nbest}")
print(f"Test set prediction ACC: "
      f"{1 - zero_one_loss(y_ts, perm_lab)}")
print(f'AMI (true labels vs predicted labels) = '
      f'{adjusted_mutual_info_score(y_ts, out.test_cllab)}')
print(f"Validation set normalized stability (misclassification):"
      f"{metrics['val'][nbest]}")
print(f'Test set ACC = {out.test_acc} '
      f'(true labels vs predicted labels)')
```

We obtain:

```
Best number of clusters: 5
Test set prediction ACC: 1.0
AMI (true labels vs predicted labels) = 1.0
Validation set normalized stability (misclassification):(0.0, (0.0, 0.0))
Test set ACC = 1.0 (true labels vs predicted labels)
```

The model perfectly identifies the 5 clusters with 0.0 misclassification error. The accuracy on test set is equal to 1.0 and the validation set normalized stability score is equal to 0.0. The prediction accuracy, as well as the AMI, obtained comparing true labels with the clustering labels returned by the model is equal to 1.0. We also observe how the choice of k-means and KNN classifier lead to the minimum stability possible, i.e., 0.0 hence satisfying Eq. (5).

Cross-validation performance is reported in Figure 3. True class labels and stability-based clustering labels when generalized to the test set are compared in Figure 4.

```
plot_metrics(metrics, title="Reval performance blobs dataset",
             legend_loc=2)
plt.figure(figsize=(6, 4))
plt.scatter(X_ts[:, 0], X_ts[:, 1],
            c=y_ts, cmap='rainbow_r')
plt.title("Test set true labels (blobs dataset)")
plt.show()


plt.figure(figsize=(6, 4))
plt.scatter(X_ts[:, 0], X_ts[:, 1],
            c=perm_lab, cmap='rainbow_r')
plt.title("Test set clustering labels (blobs dataset)")
plt.show()
```

Real-world dataset

For the real-world dataset example, we load the hand-written digits dataset [21] that includes 1797 samples corresponding to 8×8 images of digits from 0 to 9. When flattened, each sample has 64 features. The dataset has 10 classes, with ~ 180 samples for each class.

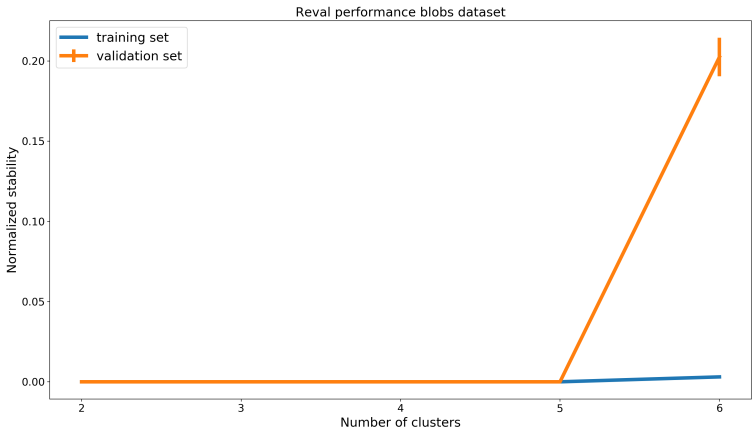


Figure 3: **reval** performance for the blobs dataset. Orange line represents the validation normalized stability with 95% confidence intervals. Blue line reports the training stability.

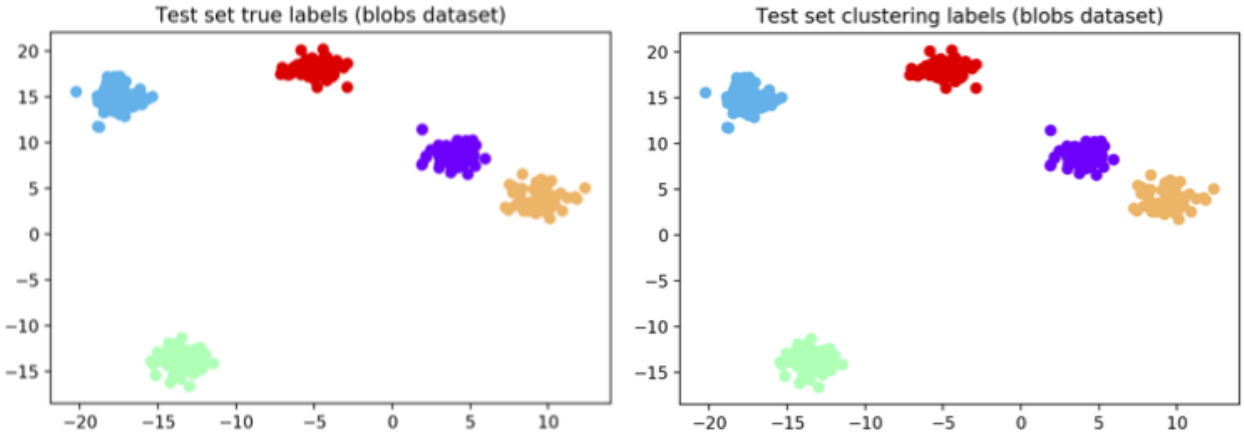


Figure 4: True classes (left) and predicted classes (right) for blobs test set after uniform manifold approximation and projection (UMAP) preprocessing.

First, we split the dataset into training and test sets of 1078 and 719 samples, respectively (40%). Then, we preprocess the dataset with UMAP [22] for dimensionality reduction (see `https://umap-learn.readthedocs.io/en/latest/clustering.html` for parameters selection). The number of UMAP components is set to 2. Then, we initialize KNN with number of neighbors equal 30 and k-means clustering with Euclidean distance. The relative clustering validation procedure is run with 10 repetitions of 5-fold cross validation, number of clusters ranging from 2 to 14, and number of random labeling iterations equal to 100. We obtain that **reval** correctly identifies 10 clusters with misclassification error equal 0.007, hence stable solution. Accuracy on test set is equal 0.87 and AMI = 0.85. See performance curves in Figure 5 and the comparison between true labels distributions and clustering label distributions in Figure 6.

```
from sklearn.datasets import load_digits
```

```
digits_dataset = load_digits()

digits_data = digits_dataset['data']
digits_target = digits_dataset['target']

X_tr, X_ts, y_tr, y_ts = train_test_split(digits_data,
                                          digits_target,
                                          test_size=0.40,
                                          random_state=42,
                                          stratify=digits_target)

transform = UMAP(n_components=2,
                 random_state=42,
                 n_neighbors=30,
                 min_dist=0.0)
X_tr = transform.fit_transform(X_tr)
X_ts = transform.transform(X_ts)

s = KNeighborsClassifier(n_neighbors=30)
c = KMeans()

reval = FindBestClustCV(s=s,
                        c=c,
                        nfold=5,
                        nclust_range=[2, 15],
                        nrand=100)

metrics, nclustbest, _ = reval.best_nclust(X_tr, iter_cv=10,
                                           strat_vect=y_tr)

plot_metrics(metrics, title='Reval performance digits dataset')

out = reval.evaluate(X_tr, X_ts, nclust=nclustbest)
```

```
perm_lab = kuhn_munkres_algorithm(y_ts, out.test_cllab)


print(f"Best number of clusters: {nclustbest}")
print(f"Test set prediction ACC: "
      f"{1 - zero_one_loss(y_ts, perm_lab)}")
print(f'AMI (true labels vs predicted labels) = '
      f'{adjusted_mutual_info_score(y_ts, out.test_cllab)}')
print(f"Validation set normalized stability (misclassification):"
      f"{metrics['val'][nclustbest]}")
print(f'Test set ACC = {out.test_acc} '
      f'(true labels vs predicted labels)')


plt.figure(figsize=(6, 4))
plt.scatter(X_ts[:, 0],
            X_ts[:, 1],
            c=y_ts, cmap='rainbow_r')
plt.title("Test set true labels (digits dataset)")
plt.show()


plt.figure(figsize=(6, 4))
plt.scatter(X_ts[:, 0],
            X_ts[:, 1],
            c=perm_lab, cmap='rainbow_r')
plt.title("Test set clustering labels (digits dataset)")
plt.show()

Best number of clusters: 10
Test set prediction ACC: 0.87
AMI (true labels vs predicted labels) = 0.85
Validation set normalized stability (misclassification):(0.0067, (0.0067, 0.0073))
Test set ACC = 0.99 (true labels vs predicted labels)
```

Similar to the blobs example, also for the real-world example we observe how the choice of k-means and KNN classifier lead to a solution close to the minimum stability possible, as requested by the model.

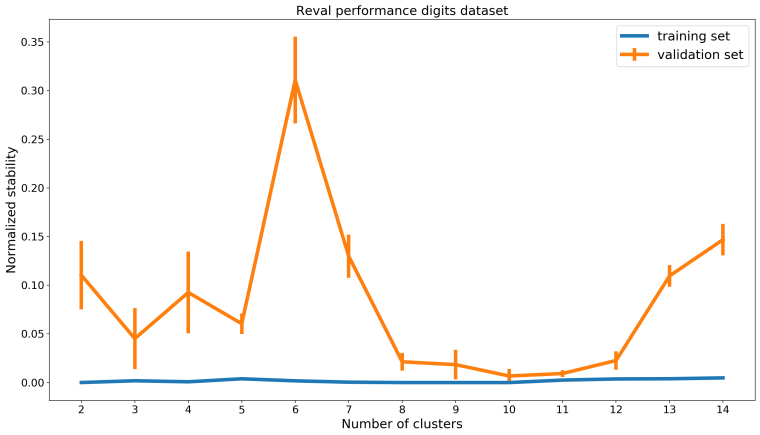


Figure 5: **reval** algorithm performance for the hand-written digits dataset.

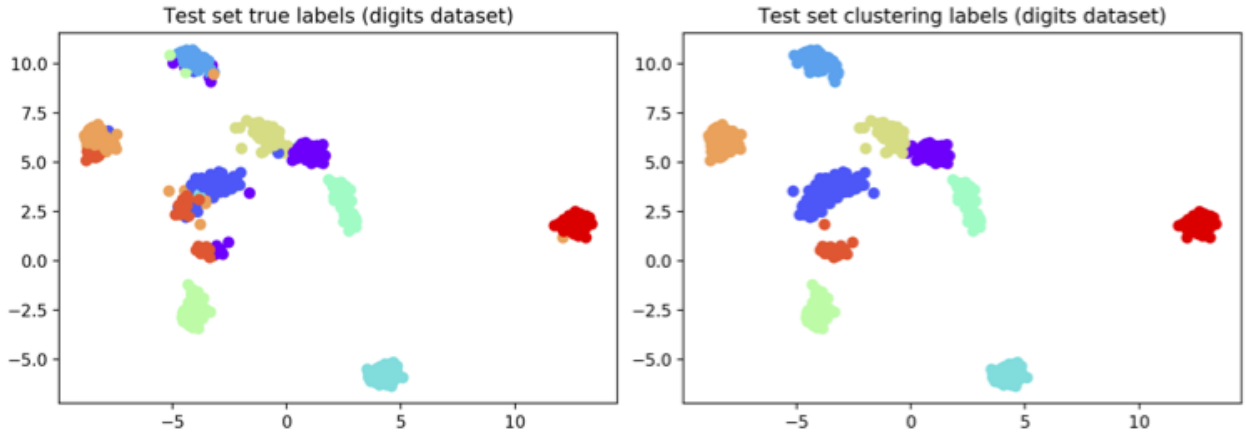


Figure 6: True classes (left) and predicted classes (right) for the hand-written digits test set after uniform manifold approximation and projection (UMAP) preprocessing.

Stability regime to guide cluster selection

Ben-Hur and colleagues presented a stability-based method with data sub-sampling and reported on the risk of underestimating/overestimating the true number of clusters [8]. Introducing prediction strength computed via repeated cross validation, Tibshirani et al. [11] linked unsupervised to supervised learning in the attempt to overcome the best cluster estimation issue. The **reval** implementation moves forward adding the evaluation of the best solution on unseen data, an approach particularly suitable for dataset with big sample size, which are becoming widely available. This feature can be leveraged to select the best number of clusters not only based on validation metrics (e.g., prediction strength greater than 0.8/0.9 [11]), but also on the performance of the solution applied to a new set of data. In this way, we are able to compare test set distribution to the one on which the result was based, hence reinforcing the decision. Selecting the number of clusters that best generalizes to new data (i.e., investigation of the stability regime) holds promise for

overcoming the underestimation issue. To give a better sense of how this works in practice we present a case study in autism research where a less conservative clustering solution results more promising when the model is evaluated on the test set.

Autism spectrum conditions (ASCs) are characterized by impairment in social communication, interaction and by restricted and repetitive behaviors [23]. Although convergent at the diagnostic level, autistic individuals may report different levels of severity and comorbidities [24]. This suggests that ASC is a heterogeneous condition with multiple subgroups tied to diverse genetic and environmental etiologies that show a range of clinical phenotypic characteristics under behavioral, linguistic and cognitive levels. To facilitate diagnosis and intervention it becomes crucial to identify autism subgroups at different levels of analysis (e.g., behavior, genetics) [25].

The national database for autism research (NDAR) dataset (`https://nda.nih.gov/`) includes a heterogeneous collection of de-identified human subjects data from autism research. We focus here on behavioral data and in particular on adaptive behavior measures collected from autistic individuals who were administered the Vineland adaptive behavior scales [26, 27]. We train the stability-based model on 420 subjects (mean age 41.65 (17.91) months, female/male counts 109/311) including the instrument domain scores (i.e., communication, living skills, socialization) after UMAP preprocessing. We run 100×3 repeated cross validation and 100 iterations of random labeling. The number of clusters ranges from 2 to 10, clustering algorithm is k-means, and classifier is KNN with number of neighbors equal 15.

From the performance plot in Figure 7 we can observe that both 2-cluster and 3-cluster solutions report small stability (2-cluster solution (error): 0.027 (0.004); 3-cluster solution (error): 0.036 (0.003)). Based on the minimization of the normalized stability measure, the algorithm selects 2 as the best number of clusters. However, if we evaluate these solutions on the test set, which includes 344 subjects (mean age 43.12 (17.04) months, female/male counts 90/254), we observe that both reach 94% accuracy. Moreover, it is straightforward to observe (see Figure 8) that the cluster in the middle (purple) of the 3-cluster solution (Figure 8b) is separated into two distinct clusters in the 2-cluster solution (Figure 8a). The generalization performance and the visual inspection indicate that we are possibly underestimating the true number of clusters [11]. In conclusion, based on the stability regime, we could select 3 as the true number of clusters: although it is a less stable solution than a more conservative number of clusters, it still generalizes well.

Summary and discussion

Although a thorough theoretical and experimental analysis of stability-based model selection with k-means clustering have been done [7, 8, 3], the effectiveness of such an approach needs to be further investigated

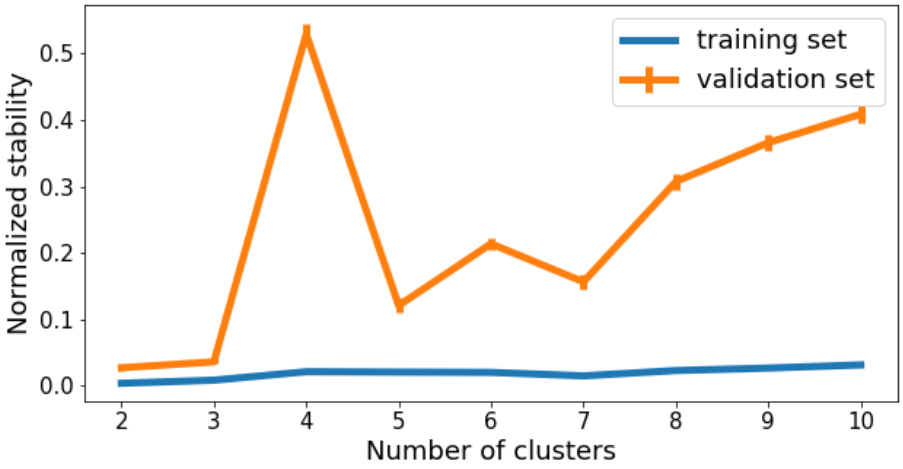
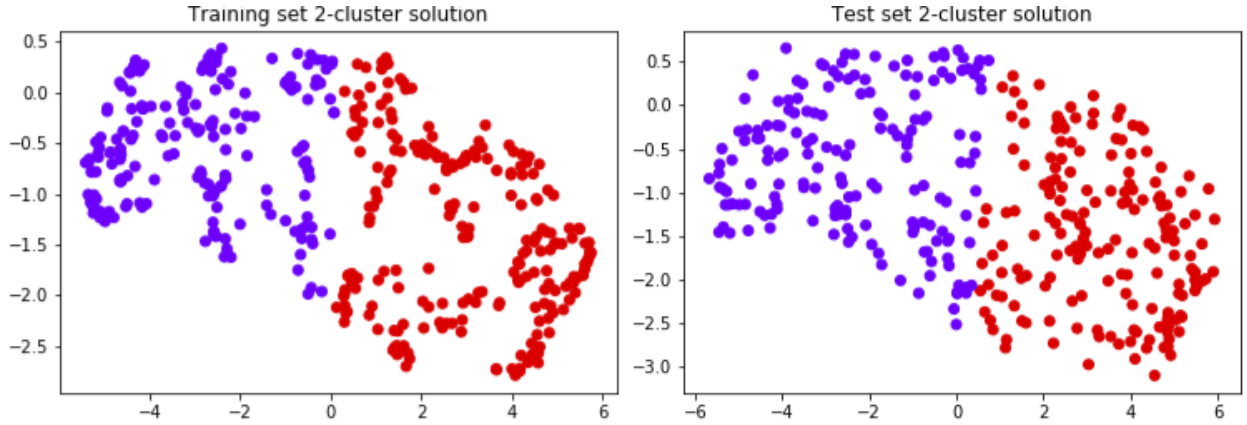
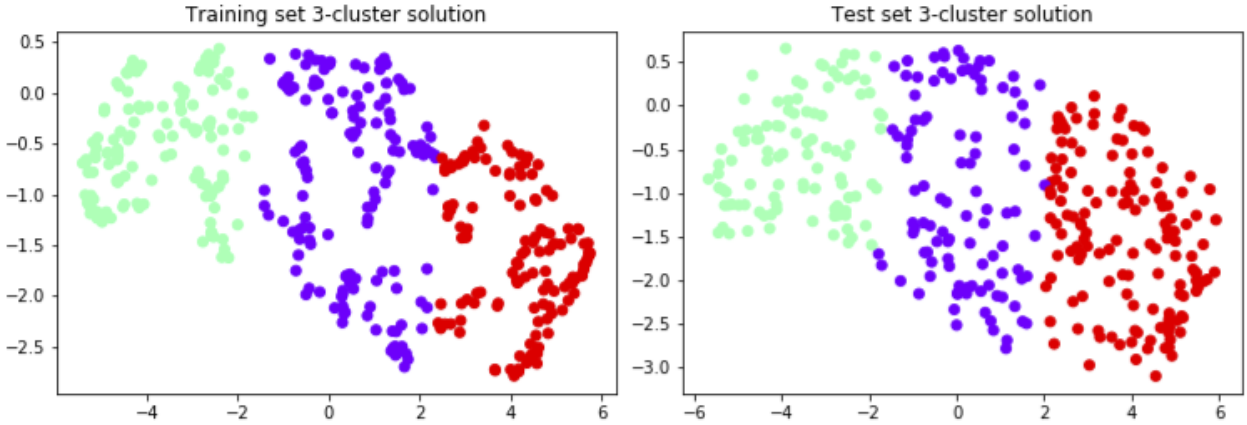


Figure 7: Stability regime for the national database for autism research (NDAR) dataset.



(a) 2-cluster solution for NDAR training (left) and test (right) sets.



(b) 3-cluster solution for NDAR training (left) and test (right) sets.

Figure 8: Clustering solution comparisons for the national database for autism research (NDAR) dataset. Data is preprocessed with uniform manifold approximation and projection (UMAP) for visualization.

with different clustering algorithms. Because **reval** works with multiple clustering algorithms, it can foster a more thorough investigation of clustering mechanisms other than k-means. Furthermore, it can be included

in ensemble learning pipelines [20] or integrated in an ensemble clustering framework for the selection of the best clustering solution [28]. In addition, in order to avoid biased estimates of the true number of clusters that underlies a distribution, we enabled **reval** with the possibility to evaluate a clustering solution on new data. This allows users to select less/more conservative solutions through the investigation of their stability regimes.

In this work, we have presented the **reval** package and how it works along with its performances in simulations, where the implemented stability-based algorithm successfully identified the correct number of clusters of most of the real-world input datasets. However, it is worth acknowledging, that because the method is based on data sampling, increasing the number of samples to preserve class distributions may be of help to detect the true number of clusters. Moreover, data preprocessing steps might be fundamental in order to find the true cluster partition underlying a data set. Examples for such scenarios can be found at `https://reval.readthedocs.io/en/latest/datadimension.html`.

Computational details and code availability

Code, package installation instructions, and documentation with different working examples using hierarchical clustering and KNN, can be found at `https://github.com/IIT-LAND/reval_clustering`. The code to replicate the simulations presented in the Simulations section is in `working_examples.manuscript_examples.py`. Code was written in Python 3.8 and simulations were run on a MacBook Pro (16-inch, 2019) with a 2,6 GHz 6-Core Intel Core i7 processor.

The code has been optimized to speed up computations. However, it is worth acknowledging that at each cross-validation iteration we: 1) apply two clustering algorithms, one to each fold; 2) fit a classifier to the training set and evaluate it on the test set; 3) normalize the stability measure after estimating the stability from N iterations of random labeling. For these reasons, the computational cost tends to increase with dataset size (see Figure 9 for an example of execution time performances).

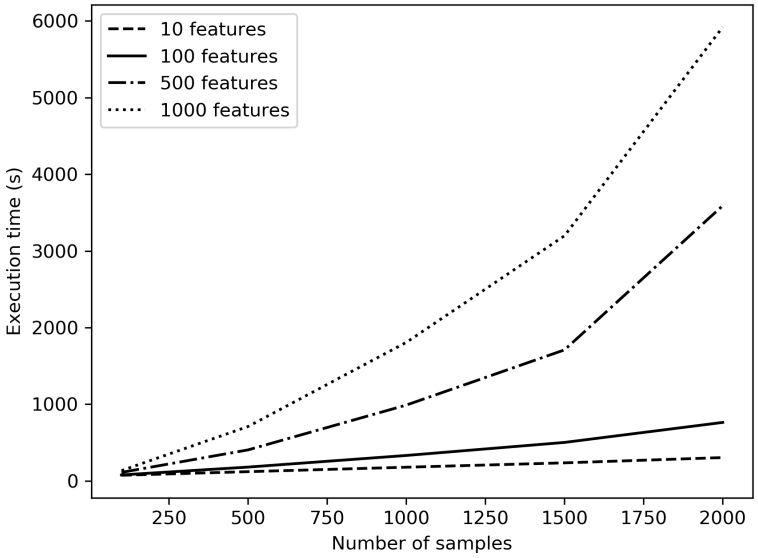


Figure 9: `best_nclust_cv` module applied to simulated blobs data with 5 clusters and varying number of samples and features. Number of clusters ranges from 2 to 6, number of repeated cross validation is set at 10×10, and number of labeling iterations is set at 100. We report execution time in seconds for algorithm performance.

Acknowledgments

This work was supported by an ERC Starting Grant (ERC-2017-STG; 755816).

Author contributions

Conceptualization, I.L. and M.V.L; Methodology, I.L.; Software, I.L. and V.M.; Investigation, I.L. and V.M.; Writing – Original Draft, I.L. and M.V.L.; Writing – Review & Editing, I.L., V.M., and M.V.L.; Funding Acquisition, M.V.L.; Supervision, M.V.L.

Declaration of interests

The authors declare no competing interests.

References

[1] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[2] Michalis Vazirgiannis. "Clustering Validity". In: *Encyclopedia of Database Systems*. Ed. by Ling Liu and M. Tamer Özsu. Boston, MA: Springer US, 2009, pp. 388–393. DOI: 10.1007/978-0-387-39940-9_616.

[3] Tilman Lange et al. "Stability-based validation of clustering solutions". In: *Neural computation* 16.6 (2004), pp. 1299–1323.

[4] Robert L Thorndike. "Who belongs in the family". In: *Psychometrika*. Citeseer. 1953.

[5] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[6] Malika Charrad et al. "NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set". In: *Journal of Statistical Software* 61.6 (2014), pp. 1–36.

[7] Ulrike von Luxburg. "Clustering Stability: An Overview". In: *Foundations and Trends® in Machine Learning* 2.3 (2010), pp. 235–274. ISSN: 1935-8237. DOI: 10.1561/2200000008.

[8] Asa Ben-Hur, Andre Elisseeff, and Isabelle Guyon. "A stability based method for discovering structure in clustered data". In: *Biocomputing 2002*. World Scientific, 2001, pp. 6–17.

[9] Ulrich Möller and Dörte Radke. "A cluster validity approach based on nearest-neighbor resampling". In: *18th International Conference on Pattern Recognition (ICPR 06)*. Vol. 1. IEEE. 2006, pp. 892–895.

[10] Nguyen Xuan Vinh and Julien Epps. "A novel approach for automatic number of clusters detection in microarray data based on consensus clustering". In: *2009 Ninth IEEE International Conference on Bioinformatics and BioEngineering*. IEEE. 2009, pp. 84–91.

[11] Robert Tibshirani and Guenther Walther. "Cluster validation by prediction strength". In: *Journal of Computational and Graphical Statistics* 14.3 (2005), pp. 511–528.

[12] Marcel Brun et al. "Model-based evaluation of clustering validation measures". In: *Pattern recognition* 40.3 (2007), pp. 807–824.

[13] Benjamin Bengfort and Rebecca Bilbro. "Yellowbrick: Visualizing the Scikit-Learn Model Selection Process". In: *The Journal of Open Source Software*. 1075th ser. 4.35 (2019). DOI: 10.21105/joss.01075.

[14] Guy Brock et al. "clValid: An R Package for Cluster Validation". In: *Journal of Statistical Software* 25.4 (2008), pp. 1–22.

[15] Jonas M. B. Haslbeck and Dirk U. Wulff. *cstab: Selection of Number of Clusters via Normalized Clustering Instability*. R package version 0.2-2. 2018. URL: https://CRAN.R-project.org/package=cstab.

[16] Jonas M. B. Haslbeck and Dirk U. Wulff. *Estimating the Number of Clusters via Normalized Cluster Instability*. Preprint at https://arxiv.org/abs/1608.07494. 2016.

[17] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[18] James Munkres. "Algorithms for the assignment and transportation problems". In: *Journal of the society for industrial and applied mathematics* 5.1 (1957), pp. 32–38.

[19] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[20] Jorge Rodríguez et al. “Cluster validation using an ensemble of supervised classifiers”. In: *Knowledge-Based Systems* 145 (2018), pp. 134–144.

[21] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[22] Leland McInnes et al. “UMAP: Uniform Manifold Approximation and Projection”. In: *Journal of Open Source Software* 3.29 (2018), p. 861. DOI: `10.21105/joss.00861`.

[23] American Psychiatric Association. *Diagnostic and Statistical Manual of Mental Disorders*. 5th ed. 2013. DOI: `10.1176/appi.books.9780890425596`.

[24] M. C. Lai, M. V. Lombardo, and S. Baron-Cohen. “Autism”. In: *Lancet* 383 (9920 Mar. 2014), pp. 896–910. ISSN: 1474-547X. DOI: `10.1016/S0140-6736(13)61539-1`.

[25] Michael V Lombardo, Meng-Chuan Lai, and Simon Baron-Cohen. “Big data approaches to decomposing heterogeneity across the autism spectrum”. In: *Molecular psychiatry* 24.10 (2019), pp. 1435–1450.

[26] Sara S. Sparrow, David A. Balla, and Domenic V. Cicchetti. *Vineland II: Vineland adaptive behavior scales*. Circle Pines, MN: AGS Publishing, 2005.

[27] Sara S Sparrow, Domenic V Cicchetti, and Celine A Saulnier. *Vineland-3: Vineland adaptive behavior scales*. PsychCorp, 2016.

[28] Alexander Strehl and Joydeep Ghosh. “Cluster ensembles—a knowledge reuse framework for combining multiple partitions”. In: *Journal of machine learning research* 3.Dec (2002), pp. 583–617.