Кластеризация при помощи пустых кластеров

Генрих Пеникас[1], **Юрий Феста**[2]

[1] Банк России,
г. Москва, Россия
[2] Сбербанк,
г. Москва, Россия

Информация о статье
Поступила в редакцию:
18.03.2024
Принята к опубликованию:
17.06.2024

JEL C32, E43, E58, F31, F40, F41, E31, F45, G01, G21

Ключевые слова:
кластеризация, кластерный анализ, пустые кластеры, заполнение пропусков, машинное обучение.

Keywords:
clustering, cluster analysis, empty clusters, imputations, machine learning.

Аннотация
Кластерный анализ широко используется в различных научных и практических областях, связанных с анализом данных. Это важный инструмент для решения задач в таких областях, как машинное обучение, обработка изображений, распознавание текста и т.д. Отсутствие наблюдений не всегда означает отсутствие информации, поэтому предполагается, что наличие пробелов в данных, наличие "пустых" кластеров, также несёт в себе информацию об объекте исследования, как и реальные наблюдения. В этом исследовании предполагается, что мы не наблюдаем не только переменную, но и целый набор объектов, образующих отдельный кластер. Таким образом, предполагается, что отсутствующее в данных — это не факт отсутствия кластера объектов как такового, а потенциально существующие объекты, которые отсутствуют в нашей выборке. Предлагается алгоритм для определения потенциальных "пустых" кластеров для одномерных и двумерных наборов данных, учитывая их размер и расположение в пространстве признаков в зависимости от исходного распределения выборок. Реализован метод заполнения этих пробелов и оценки смещения центроидов начальной кластеризации при учёте пустого кластера. Продемонстрировано применение этого подхода для удаления выбросов из данных.

DOI: https://doi.org/10.24866/2311-2271/2024-2/1132.
Ссылка для цитирования. Penikas H.I., Festa Yu.Yu. Clistering with Empty Clusters // Известия Дальневосточного федерального университета. Экономика и управление. 2024. № 2 (110). С. 75–94. — DOI: 10.24866/2311-2271/2024-1/1132.

Clustering with Empty Clusters

Henry I. Penikas, Yury Yu. Festa

Abstract

Cluster analysis is widely used in various scientific and practical fields related to data analysis. It is an important tool for solving problems in such areas as machine learning, image processing, text recognition, etc. The absence of observations is not always the absence of information, therefore it is assumed that the presence of gaps in the data, the presence of "empty" clusters, also carries information about the object of study, as well as real observations. In this study, it is assumed that we do not observe not only a variable, but a whole set of objects forming a separate cluster. Thus, it is assumed that the missing in data is not the fact of the missing of a cluster of objects as such, but potentially existing objects that are absent from our selection. An algorithm is proposed to determine potential "empty" clusters for one-dimensional and two-dimensional data sets, taking into account their size and location in the feature space, depending on the initial distribution of samples. A method is implemented to fill in these gaps and estimate the displacement of the centroids of the initial clustering when taking into account an empty cluster. The application of this approach to rid the data of outliers is demonstrated.

1. Introduction

We come across empty clusters more often than imagine. Sometimes they are easily predefined when we have integer values for the classifying indicators. However, when the classifying attributes may take on floating values, thinking of empty clusters is not easy.

Let us illustrate the problem setting with the easiest case. Suppose we have relative grading for students. Relative means that each student earns a score during one's education term. When compared to the peers the teacher derives the ultimate mark. Suppose the score ranges from 0 to 10, so does the final mark. Let the teacher receive a score concentration similar to Figure 1. If one tried to cluster observations, most probably one is to end with two options: two or six clusters.

The major shortcoming here is that the teacher is expected to grade the person having a score of 5 just one mark lower, than the student having a score of 8, because 5 and 8 obviously belong to different clusters. However, we see that there is large gap (unconcentrated points, *missings*) on the horizontal axis for the scores of 6 and 7. This means that the student with 5 scores in our case should be graded with 5 as a mark and not 7 if we grade 8 scores with a mark of 8. However, none of the existing clustering procedures allows an ML practitioner to identify these empty clusters in around 6 and (or) 7 from Figure 1. We intend to start closing this gap.

The purpose of the work is to propose an algorithm that performs clustering for *a priori* a given number of clusters, demonstrates an area from an "empty" cluster and conducts the procedure for implementing a new cluster. That is why our objective is to start from the baseline case and suggest an algorithm that runs one-dimensional and two-dimensional clustering for a given number of clusters and demonstrates the domain of an empty cluster.

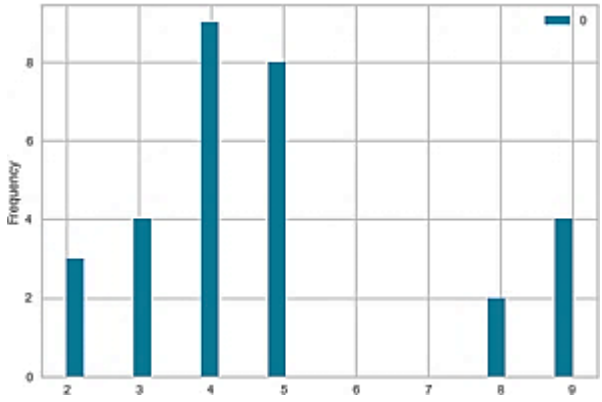


Figure 1: Simulated Data Without Noise

We structure our paper as follows, Section 2 reviews the literature. Section 3 describes the simulated (artificial) dataset. Section 4 presents the suggested methodology. The findings are reported in Section 5. Section 6 concludes.

2. Literature review

It is worth discussing the progress in two areas prior to identifying the empty clusters. First, we think of the probable nature of missing data in subsection 2.1. Second, we describe the accumulated knowledge on how to treat empty clusters in subsection 2.2.

2.1. ***Missing data types: MCAR, MAR, MNAR*** every data scientist comes across missing data in their studies. The generally accepted classification of missing data has been proposed by Rubin (1976). It proposes a classification of missing data based on the probabilistic nature of these missings. Missing data is classified as **MCAR** (missing completely at random) if missing occurs as a result of events that lead to the systematic absence of any particular element in the data. They are independent of the observed parameters and variables and occur completely randomly, i.e., the probability of missing for each element in the data is the same and gaps must be distributed randomly in the data.

Missing data is classified as **MAR** (missing at random) when missing elements in data arrays are not completely random, but arise due to certain patterns. The probability of missing an element can be determined based on other information available in the set that does not contain omissions. The exclusion or replacement of a missing element with a certain value does not lead to a significant distortion of the results and can be fully explained by variables for which complete information is available, that is, calculated relative to other features with varying degrees of confidence.

If the missing data does not fall under the previous categories, then we are dealing with **MNAR** (missing not at random). In this case, we have a systematic omission of a certain category in the data. This may occur as a result of incorrectly conducted social surveys or questionnaires in medicine, as mentioned by Pereira et al. (2019). There are no observations

depending on unknown factors, data omissions are systematically associated with their expected values. It is assumed that the probability of missing cannot be expressed based on the information contained in the dataset.

Carreras et al. (2021) points out that in the case of MCAR, it is possible not to use imputations, since the absence of data is completely random and does not lead to a bias in estimates and model parameters. The main difference between MNAR and MAR is that missing data are related to unobserved data and it is impossible to evaluate or estimate missing data and all imputation methods not valid for MNAR. However, not all authors believe that in MNAR case it is incorrect to apply imputations, see for example Heymans and Twisk (2022). The only way to get an unbiased estimate of the parameters in such a case is to model the missing data.

In this article, it is assumed that an entire cluster of objects, which we call an empty cluster, is not included in our dataset and this missing is classified as MNAR. We also assume that this empty cluster carries useful information and ignoring these omissions may lead to an incorrect interpretation of the source data or a bias in the estimation of model parameters. It is assumed that there is no whole cluster in the dataset or only a certain number of observations from it. In this case, outliers in the data may not mean "unnecessary observations" at all, but a new cluster that we do not observe. For example, participants in a sociological study may refuse to provide information or not have access to surveys for various reasons, or when conducting a study, the respondents may be divided into subsamples for more effective analysis, while the researchers did not take into account any specific category of respondents.

In economic research, there are limitations on the amount of data collected due to the non- timely provision of reporting data, or the lack of data in the source when collecting information. For example, in credit risk tasks, an incorrect initial definition of a borrower's cluster may lead to an error in estimating the probability of a borrower's default. Thus, this approach changes not only the interpretation of outliers, but also allows for re-evaluation of clustering output.

2.2. *Handling empty clusters*

Historical note. Cluster analysis has nowadays become a popular area of instrumental or explanatory data analysis. By today it is more than 60 years old when counting from one of the first works by Robert Sokal and Peter Sneath in Sokal and Sneath (1963). It was then followed by French researcher Diday (1972, 1979), also reviewed by Sokal (1984). Edwin Diday contributed much to the systematic development of software for the cluster-analysis. To name a few recent handbooks we may cite Mirkin (2016); Raschka and Mirjalili (2019). Useful review can be found in Xu and Tian (2015) where authors consider current SOTA (State of the Art) approaches to cluster analysis and their application areas.

Empty clusters: visual identification and regression bias. Empty clusters can be explained in different ways, depending on the definition. It is indirectly mentioned by Forina et al. (2003). The authors propose a statistical test to assess the quality of agglomerative clustering, and also

propose an index of the informativeness of "empty spaces", taking into account the greatest distance between clusters. Giesen et al. (2017) developed of a visualization tool for dot scatter diagrams with the allocation of "empty" clusters.

McGee et al. (2020) investigate the long-term effects on health, in particular, on the birth rate of children, as a result of external influences and note that approaches using estimation equations necessarily exclude empty clusters and, therefore, give biased estimates of marginal effects. Some applied researchers suggest using the cluster label as an additional feature in the regression problem like Piernik and Morzy (2021). Thus, an error in clustering can lead to a bias in the regression estimates.

Initialization. The *initialization* of an empty cluster is a feature of the implementation of the initial algorithm and this case is considered as a disadvantage of the method, for example, k-means, and researchers try to minimize such cases by upgrading classical algorithms. This situation may occur when the correct initial conditions are not met and the optimizer finds a local minimum, or when the input data is represented by binary or categorical features (Raykov et al., 2016). (Raschka and Mirjalili, 2019, Ch. 11) note the disadvantages of algorithms, such as convergence, computational complexity, initialization of empty clusters.

Yadav and Dhingra (2016) and Pakhira (2009) propose modifications of the k-means algorithm, which are not initialized by empty clusters. The authors take into account the positions of the centroids and the values of gap statistics or use more modern optimizers. Hua et al. (2019) introduce another implementation of the algorithm — the genetic XK-Means, which also does not initialize empty clusters. Tavallali et al. (2021) note that the initialization of empty clusters is detected as a result of the application of the random selection algorithm.

Imputation. Another reason for the appearance of empty clusters is the incompleteness of the data: if the data does not contain values for some variables, this may lead to exclusion of the relevant observations from clustering. To minimize this effect, the *imputation* procedure is used, see Audigier et al. (2021). The idea is to first fill the data set with observations corresponding to otherwise unidentifiable clusters. The disadvantage of the approach is that it requires prior knowledge of where new observations are being added.

Summary. Though clustering specifics seems to be well studied, the challenge of empty classes seems to be unresolved. Heymans and Twisk (2022) proposed to expand the idea of *imputations* to cluster analysis in general. It is assumed that there is no whole cluster in the dataset oronly a certain number of observations from it. In this case, outliers in the data may not mean "unnecessary observations" at all, but a new cluster that we do not observe.

3. Simulated Data

We start with the one-dimensional case in subsection 3.1 and extend it to the two-dimensional one in subsection 3.2.

3.1. ***One-dimensional case***. We start from a uni-dimensional case like in Figure 2. As a robustness check we add noise. We have 30 observations overall which may take values from 2 to 9. For the two-dimensional case we use the make_blobs method from the sklearn library. These datasets can be interpreted as mixtures of normal distributions with a given mode and standard deviation. Since we know the initial number of clusters a priori, we assume that the initial number of clusters will be two, but the added noise makes it difficult to choose an explicit number of clusters. To check the optimal number of initial partitions, we will use the metrics Distortion Measure (Elbow-method), Silhouette Analysis, Calinski Harabas. K-means is used as a clustering algorithm with implementation in the sklearn library with the k-means++ initializer.

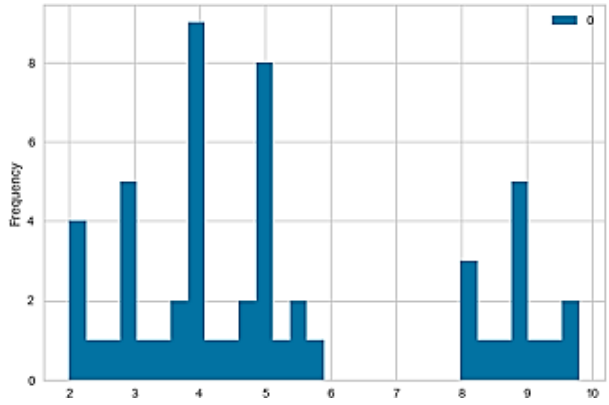
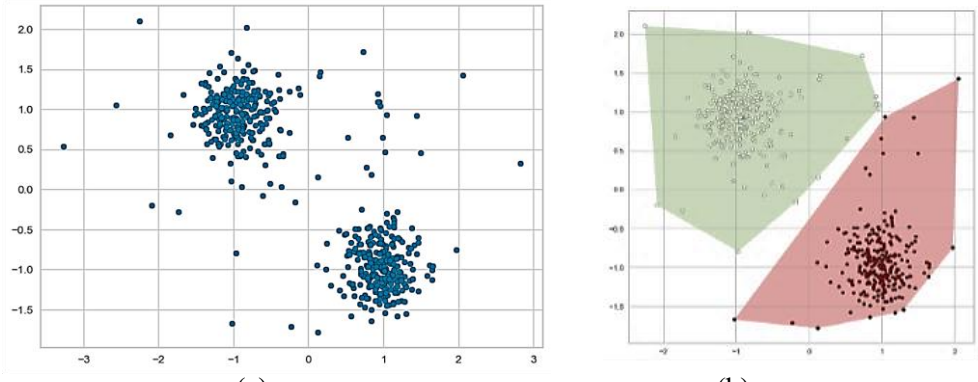


Figure 2: 1-Dimensional Simulated Data

3.2. ***Two-dimensional case***. We generate two normally distributed datasets of 150 objects. Next, we add randomly distributed noise with a conditional center between the two previous datasets. As a result, we got a mixture of normal distributions and random noise. The initial data and the results of k-means clustering for two centers are shown in Figure 3.



(a) (b)

Figure 3: Visualization of initial data (a)
and clustering results for two clusters (b)

To visualize the potential location for a latent “empty” cluster in the two-dimensional case, we use the construction of a *Voronov diagram* like in Figure 4. It can be assumed that in the area between the initial clusters, where the polygonal grid is most sparse, there is an “empty” cluster. The same procedure was made by Reddy and Jana (2012).

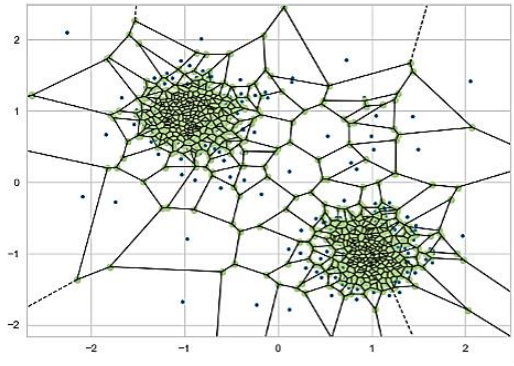


Figure 4: Voronov diagram for 2-Dimensional data

4. Methodology

We may not need searching for the optimal number of clusters if we simulated data. However, in real-life when the data generation process is unknown we would have thought of empty clusters existence in addition to the ones optimally identified by the current procedure. The procedure itself consists of the following steps:

- Estimate the range of the number of clusters for a dataset;
- Apply the clustering algorithm with the number of clusters selected in the previous step;
- For the found clusters, evaluate the distribution function of variables using the maximum likelihood method and find the distribution parameters;
- Find the geometric center between the centroids;
- Generate data from a mixture of the found distributions and place the center of the new cluster at an equidistant point between the cluster centroids;
- Restart the clustering algorithm on new data with the number of specified clusters increased by one;
- Compare cluster parameters, select points from the first partition of the dataset that have changed the cluster label;
- For the original dataset, leave the cluster labels taking into account the empty cluster.

After the procedure is completed, further investigation of the found “empty” cluster is expected. These points can be considered as outliers, and not taken into account in our data. This way we can reduce the error of the classification algorithm. On the contrary, for the remaining points of the “empty” cluster, we may try to restore the type of distribution and add them to the data again.

In general, the next steps depend on the area of specific research and the task being solved. The article discusses how the removal of these points will affect the metrics of the classification algorithm. Since the data is synthetic (simulated, artificial) and clearly linearly separatable, we will use the random forest classifier and quadratic discriminant analysis as classification algorithms.

The developed programming codes in Python are available in Annexes A.1, A.2.

5. Findings

We discuss the one-dimensional case first in subsection 5.1; two-dimensional one follows in sub-section 5.2.

5.1. ***One-dimensional case***. Figure 5 provides the visual illustration for the conventional clustering procedures. The vertical dashed line indicated the optimal number of clusters per procedure. It equals to 4, 2 and 5, respectively.

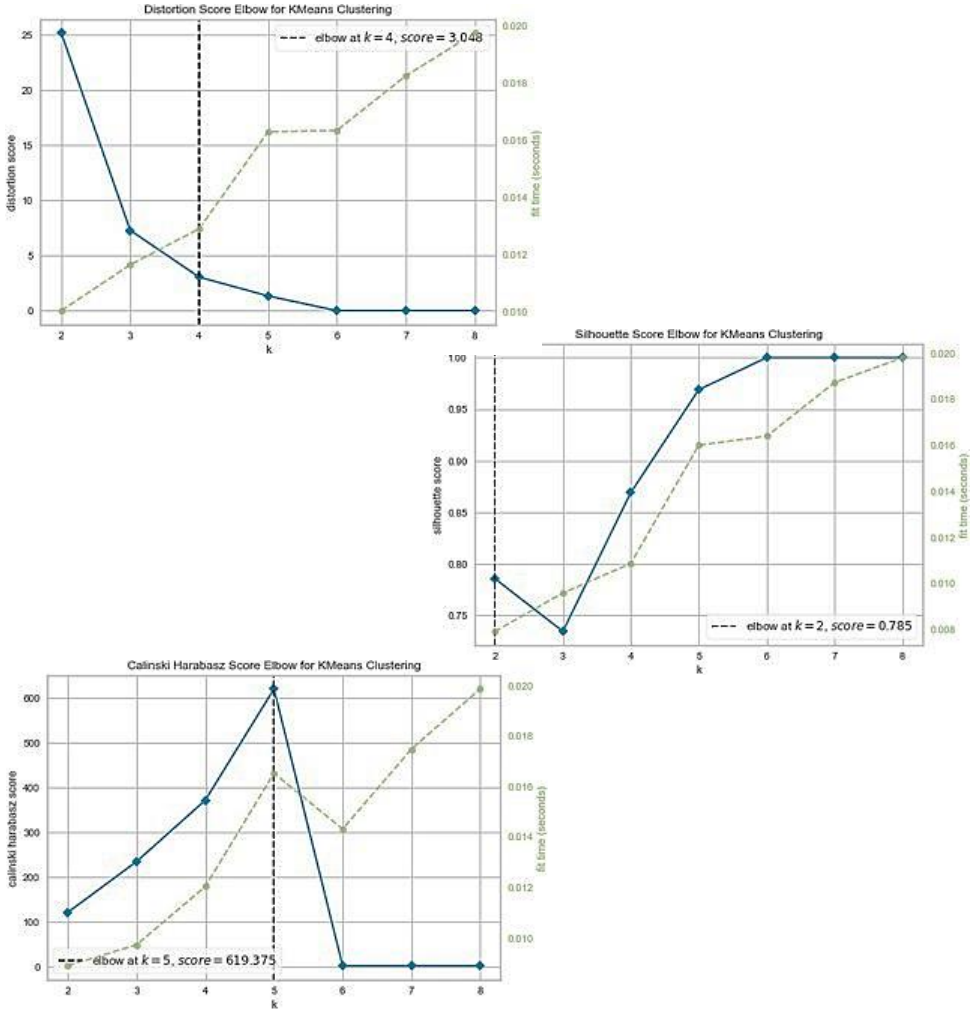


Figure 5: Clustering Data Without Noise:
top left — Distortion Measure; *top right* — Silhouette Analysis;
bottom — Calinski Harabasz

Figure 6 illustrates how the empty clusters are identified for the simulated data without noise and with it. For visibility we imputed 20 artificial observations for an empty clusters at the values of 6 and 7.

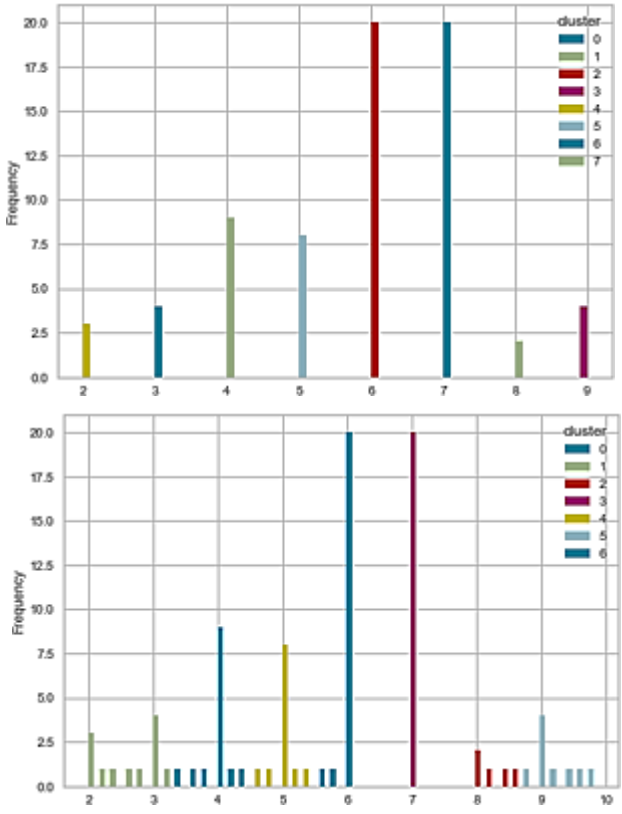


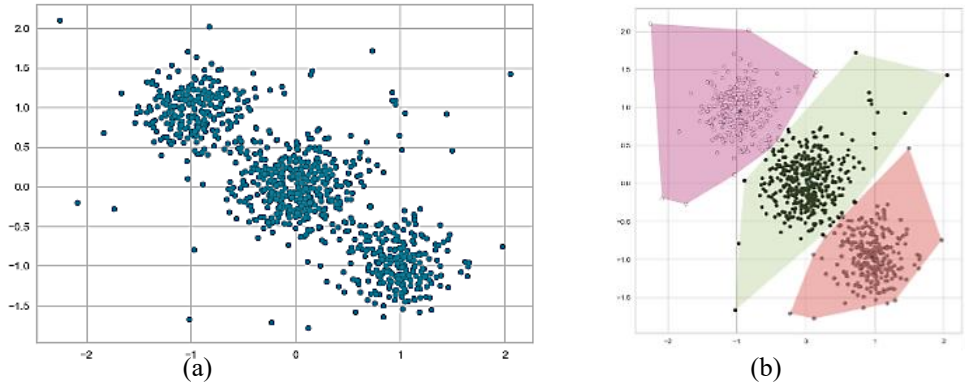
Figure 6: Identifying Empty Clusters
for Data Without Noise (left)

5.2. ***Two-dimensional case***. We define the type of distribution and its parameter for the found clusters. After that, we generate a new data set from a mixture of these distributions in proportion to the size of the original clusters and place the center of the new cluster in the geometric center between the found clusters. After that we restart clustering algorithm with three clusters. New data and results of k-means clustering for the three centers are presented in Figure 7.

Next, we remove the imputed data, but leave the cluster labels from the last iteration of the algorithm. The resulting clusterization and clusterization of the initial data set for the number of clusters equal to three are shown in Figure 8.

Thus, we have isolated the boundary points from the initial partition and assigned them to a separate cluster. In a specific example on the scatter plot, it is noticeable that the data inside the “empty cluster” is correlated (i.e., X1 is correlated to X2), although it was not initially assumed that there is a stable linear relationship between the explanatory variables. Thus, we have demonstrated that noise in the data can actually be a potential cluster.

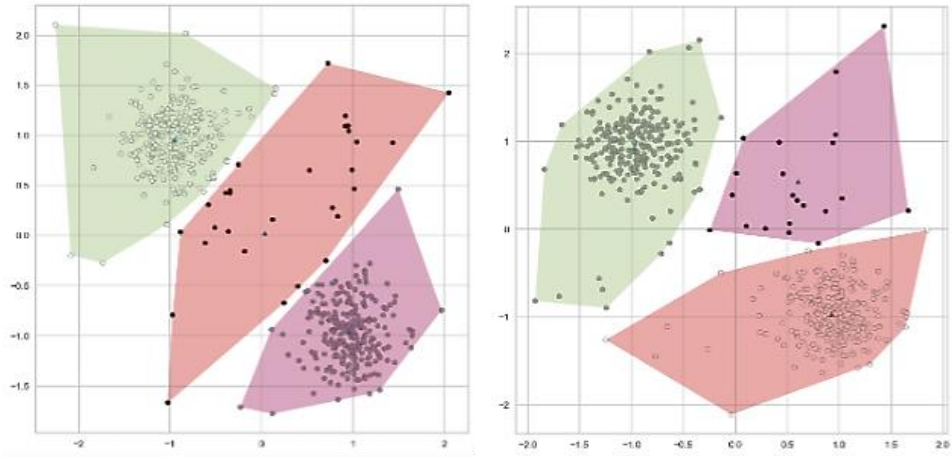
It would be a mistake not to take into account its presence when constructing classification models, since this would cause a bias in model parameters.



(a) 2-Dimensional Simulated Data with added cluster

(b) K-means clustering result with added cluster

Figure 7: Visualization of new data and clustering results for three clusters



(a) (b)

Figure 8: Visualization of the source data with an “empty” cluster and initial partitioning for three clusters

Next, consider the classification problem, where the class labels will be the cluster labels from the first iteration. Since the classes are balanced, it was enough to choose only Accuracy as the metric of the algorithm, but in this study we will also consider *Precision*, *Recall* and *F*-1 *measure*. Let us compare the metrics of the classification algorithm for two data sets: the original one and the one with the removed elements of the “empty” cluster. The data were divided into training (source) and validation (clear) samples. The following algorithms were chosen for classification: Linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), random forest classifier (RFC).

Table shows the accuracy metrics for each algorithm for the two datasets. Despite the initial, almost linear division of classes, there is an

increase in quality metrics for each algorithm in the validation sample. Thus, the "empty" cluster can also make changes to the parameters of the classification algorithm, which is especially bad when using *embeddings* (i.e., artificially derived data based on the initial dataset, might be projected values from the interim neural network layer) as explanatory variables.

The increase in metrics of classification algorithms

	Accuracy		Precision		Recall		F1	
	Source	Clear	Source	Clear	Source	Clear	Source	Clear
LDA	0,9994	1,0000	1,0000	1,0000	0,9988	1,0000	0,9994	1,0000
QDA	0,9988	1,0000	1,0000	1,0000	0,9976	1,0000	0,9988	1,0000
RFC	0,9958	0,9982	0,9941	0,9976	0,9976	0,9988	0,9959	0,9982

Note: algorithms by rows: LDA — linear discriminant analysis, QDA — quadratic discriminant analysis, RFC — random forest classifier; samples by columns: source — training, clear — validation.

6. Conclusions

We proceed with the overview of the accomplished work in subsection 6.1; we elaborate on the research extensions in subsection 6.2 and offer policy implications applicable to a commercial bank in subsection 6.3.

6.1. ***Brief summary***. The procedure with the assignment of empty clusters was already implicitly considered in the literature, though with certain limitations. Here our objective was to start from the baseline case and suggest an algorithm that runs one- and two-dimensional clustering for a given number of clusters and reveal the domain of an empty cluster.

We have proposed such an algorithm. When varying the size of an empty cluster with constant modes and standard deviation, it is noted that:

• if its size and width are less than the average of the original ones, then the centroids practically do not shift relative to the original ones;

• if equal or greater, then, on the contrary, there is a shift of the centroids and more objects receive a new label.

Thus, the algorithm for allocating and generating an empty cluster should be parametric. It is demonstrated that this algorithm can be used to clean up data from outliers when solving the classification problem, which leads to an increase in classification accuracy metrics.

6.2. ***Research extensions***. The current research has two natural extensions:

• Identify potential empty clusters not only inside the existing space, but also *outside* the boundaries (range or domain) of the original (observed available) data. Figure 1 shows the simulated values from 2 to 9 with the empty places in 6 and 7. However, one may fairly argue that the values of 1 and 10 are also empty. This means that one may extend our procedure to identifying empty clusters not only in the interior of the existing values, but at its exterior.

Here we wish to mention the useful Bayesian rule when choosing whether to assign an empty cluster in a unidimensional case: on the left or on the right. Consider again Figure 1. The data is concentrated to the left (below 5). It seems that concentration in the lower tail should imply the existence of an empty class on the left.

However, we argue that we should look right on the opposite, i.e., to the right. If the existing data is densely populated on the left, then by the probability theory the potential empty cluster on the left should have been filled already. Then we naturally assume that the extreme empty cluster on Figure 1 would lie on the right, and not on the left.

• Identifying empty clusters in the multi-dimensional setting for more than two dimensions. In the current implementation of the algorithm for multidimensional data, one can attain this by reducing the dimension space to two using data dimension reduction algorithms like PCA or t-SNE.

As a research extension it might be useful to recall the procedures for the modeling of the confidence sets (regions), as a next step after the confidence intervals' modeling. For more details on confidence regions, please, see (Shvedov, 2016, pp. 134-138, section 3.4), (Demidenko, 2019, pp. 594-596, section 7.8.5).

6.3. ***Policy implications***. When developing a business strategy for a bank, a banker may wish to segment all its borrowers into distinct clusters. Often the obtained groups reflect the creditworthiness of a bank client. However, imagine that in the future a client from an unforeseen cohort applies to a bank. Price offers based on any of the existing clusters would be inadequate. They would either be unappealing to a client and he is going to reject and move to a competitor. More troublesome is the case when the offer would not reflect the sufficiently elevated risk level. The new customer would highly likely accept the low-rated offer and the bank is to bear the extra losses some time later. The potential remedy here would be running the initial segmentation with a preview of the existence of additional (empty) clusters which are not observed in the available (historical) data. The developed code from the Annex can be used for the purpose.

A. Relevant Python Code

A.1. *One-dimensional case*

```
def add_cluster(df, model_, nclust,res1_arr):
    mins = []
    maxes = []
    diffs = []

    df_ = df[df['values'].isin([1,2,3,4,5,6,7,8,9])] for
    clust in df_.cluster.unique():
          min_ = df_[df_['cluster']==clust]['values'].min()
          max_ =      df_[df_['cluster']==clust]['values'].max()
          diff_ = max_ - min_

          mins.append(min_)
          maxes.append(max_)
          diffs.append(diff_)
```

```
    print(mins)
    print(maxes)
    print(diffs)

    gaps = []

    for i in range(0, df_.cluster.nunique()):
        if i == 0:
            gap0 = min(mins)
            gaps.append(gap0)
        else:
            gap = (mins[i] - maxes[i-1])
            gaps.append(gap)
    print(gaps)

    df_c = pd.DataFrame({'mins':mins, 'maxes':maxes, 'diffs':diffs, 'gaps':gaps})
    print(df_c)

    indd = df_c[df_c['gaps'] == df_c['gaps'].max()].index
    val_from = int(df_c.iloc[indd-1][['maxes']].values[0][0])
    val_to = int(df_c.iloc[indd][['mins']].values[0][0])

    print(val_from)
    print(val_to)
#       val_from_ind = df_c[df_c['mins']!=val_to_ind]['maxes'].max()+1
#       print(val_from_ind, val_to_ind)
#       print(df_c[df_c['gaps'] == df_c['gaps'].max()].index)

    index_to_insert = res1_arr.index(val_from)
    new_arr = res1_arr.copy()

    values_to_insert = [i for i in range(val_from+1,val_to)]
    values_to_insert = [[i]*20 for i in values_to_insert]
    values_to_insert = sum(values_to_insert, [])

    print(values_to_insert)
    for i in values_to_insert:
        new_arr.insert(index_to_insert, i)
        index_to_insert += 1

    new_arr_ = np.array(new_arr).reshape(-1,1)
    print(new_arr)
    pd.DataFrame(new_arr).plot(kind = 'hist', bins = 30)

    model = model_(n_clusters=nclust)
    model.fit(new_arr_)
    yhat = model.fit_predict(new_arr_)
    clusters = np.unique(yhat)
    df = pd.DataFrame({'cluster':yhat, 'values':new_arr})
    make_me_hist_(df)
```

A.2. Two-dimensional case

```
# imports

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from fitter import Fitter, get_common_distributions, get_distributions
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score,
classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression, ridge_regression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis,
QuadraticDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics.pairwise import euclidean_distances
from  sklearn.model_selection  import  train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_blobs
from scipy.spatial import ConvexHull
from scipy import stats

# data generation and data preparation
X1, Y1 = make_blobs(n_samples=5000,n_features=2, centers=2, random_state=42)
X1  =  StandardScaler().fit_transform(X1)
noise = np.random.normal(0, 1, [int(len(X1)*0.10), 2])
X1_noised = np.concatenate((X1, noise))

# visualisation
plt.scatter(X1_noised[:, 0], X1_noised[:, 1], marker="o", s=25, edgecolor="k")

# create dataframe
df = pd.DataFrame()
df['x1'] = X1_noised[:, 0]
df['x2'] = X1_noised[:, 1]
df  =  df[(np.abs(stats.zscore(df))  <  2.5).all(axis=1)]

# find initial centroids and clusters mark
kmeans = KMeans(n_clusters=2, random_state=0, init = 'k-means++')
df['cluster']  =  kmeans.fit_predict(df[['x1',  'x2']])
centroids = kmeans.cluster_centers_
cen_x = [i[0] for i in centroids]
cen_y = [i[1] for i in centroids]
# add to df
df['cen_x']  =  df.cluster.map({0:cen_x[0],  1:cen_x[1]})
df['cen_y']  =  df.cluster.map({0:cen_y[0],  1:cen_y[1]})

# visualisation
plt.scatter(df.x1, df.x2, c=df.cluster, marker="o", s=25, edgecolor="k")
sns.jointplot(x = df.x1, y = df.x2, kind = 'scatter', \
marginal_ticks  =  False,  hue  =  df.cluster)  #  visualization
```

```
make_kmeans_polygon_figure(df, 2) # visualisation

# find kind of distribution
f = Fitter(df[df['cluster']==0]['x1'].tolist(),\
distributions= get_common_distributions())
f.fit()
f.summary()

# find params of normal distribution
x1_mean_0 = df[df['cluster']==0]['x1'].mean()
x2_mean_0 = df[df['cluster']==0]['x2'].mean()

x1_std_0 = df[df['cluster']==0]['x1'].std()
x2_std_0 = df[df['cluster']==0]['x2'].std()

x1_mean_1 = df[df['cluster']==1]['x1'].mean()
x2_mean_1 = df[df['cluster']==1]['x2'].mean()

x1_std_1 = df[df['cluster']==1]['x1'].std()
x2_std_1 = df[df['cluster']==1]['x2'].std()
print(x1_mean_0 ,x1_std_0, x2_mean_0, x2_std_0)
print(x1_mean_1 ,x1_std_1, x2_mean_1, x2_std_1)

# find mean size of clusters
df.groupby('cluster').count()['x1']
df.groupby('cluster').count()['x1'].mean()

# find centeroid of "empty" cluster
new_centroid = tuple(np.median(centroids, axis=0))
print(new_centroid)

# create "empty" cluster
noise = np.random.normal(0, 1, [int(len(X1)*0.10), 2])
X1_add, Y1_add = make_blobs(n_samples=407, n_features=2, centers = 1,\
center_box = new_centroid , cluster_std = 0.32, random_state=42)

# concat initial data frames with"empty" cluster
df_old = df[['x1', 'x2', 'cluster']].copy()\
.rename(columns = {'cluster':'cluster_old'})
df_with_added = pd.DataFrame()
df_with_added['x1'] = X1_add[:, 0]
df_with_added['x2'] = X1_add[:, 1]
df_with_added = df_with_added[(np.abs(stats.zscore(df_with_added)) < 2.5)\
.all(axis=1)]
df_with_added = df_with_added.append(df_old)

# visualisation
plt.scatter(df_with_added.x1, df_with_added.x2, marker="o", s=25, edgecolor="k")

# concat initial data frames with"empty"
cluster # find new centroids and clusters mark
```

```
kmeans = KMeans(n_clusters=3, random_state=0, init = 'k-means++')
df_with_added['cluster_new'] = kmeans.fit_predict(df_with_added[['x1', 'x2']])

# get centroids
centroids = kmeans.cluster_centers_
cen_x = [i[0] for i in centroids]
cen_y = [i[1] for i in centroids]

df_with_added['cen_x'] = df_with_added.cluster_new.map({0:cen_x[0],\
1:cen_x[1], 2:cen_x[2]})
df_with_added['cen_y'] = df_with_added.cluster_new.map({0:cen_y[0],\
1:cen_y[1], 2:cen_x[2]})

# polygonal plot

fig, ax = plt.subplots(1, figsize=(8,8))
plt.scatter(df_with_added.x1, df_with_added.x2, c=df_with_added.cluster_new,\
marker="o", s=25, edgecolor="k")
plt.scatter(cen_x, cen_y, marker="^", s=25, edgecolor="k")
for i in df_with_added.cluster_new.unique():
    points = df_with_added[df_with_added.cluster_new == i][['x1', 'x2']].values
    hull = ConvexHull(points)
    x_hull = np.append(points[hull.vertices,0],
                       points[hull.vertices,0][0])

    y_hull = np.append(points[hull.vertices,1],
                       points[hull.vertices,1][0])
    plt.fill(x_hull, y_hull, alpha=0.3)

# remove "empty" cluster
df_with_added_remove_new = df_with_added[df_with_added['cluster_old'].isna()==False]

# polygonal plot with new cluster mark
fig, ax = plt.subplots(1, figsize=(8,8))
plt.scatter(df_with_added_remove_new.x1, df_with_added_remove_new.x2, \
c=df_with_added_remove_new.cluster_new, marker="o", s=25, edgecolor="k")
plt.scatter(cen_x, cen_y, marker="^", s=25, edgecolor="k")
for i in df_with_added_remove_new.cluster_new.unique():
   points = df_with_added_remove_new[df_with_added_remove_new.cluster_new == i]\
    [['x1', 'x2']].values
    hull = ConvexHull(points)
    x_hull = np.append(points[hull.vertices,0],
                       points[hull.vertices,0][0])

    y_hull = np.append(points[hull.vertices,1],
                       points[hull.vertices,1][0])
    plt.fill(x_hull, y_hull, alpha=0.3)

# metrics evaluation
df_with_added_remove_new['remove'] = df_with_added_remove_new['cluster_old']\
== df_with_added_remove_new['cluster_new']
df_new = df_with_added_remove_new[df_with_added_remove_new['remove']==True]
```

```
x_train_src, x_test_src, y_train_src, y_test_src = train_test_split(df[['x1','x2']],\
df[['cluster']], test_size=0.3, random_state=0)
x_train_new, x_test_new, y_train_new, y_test_new = train_test_split(df_new[['x1','x2'
df_new[['cluster_old']], test_size=0.3, random_state=0)

model_old = LinearDiscriminantAnalysis()
model_old.fit(x_train_src,   y_train_src)
res_src  =  model_old.predict(x_test_src)

accuracy_src =  accuracy_score(y_test_src, res_src)
precision_src = precision_score(y_test_src, res_src)
recall_src = recall_score(y_test_src,   res_src)
f1_src = f1_score(y_test_src, res_src)
model_new = LinearDiscriminantAnalysis()
model_new.fit(x_train_new, y_train_new)
res_new = model_new.predict(x_test_new)

accuracy_new = accuracy_score(y_test_new, res_new)
precision_new = precision_score(y_test_new, res_new)
recall_new = recall_score(y_test_new,   res_new)
f1_new = f1_score(y_test_new,  res_new)

print('accuracy_src: ', accuracy_src)
print('precision_src: ', precision_src)
print('recall_src: ', recall_src)
print('f1_src: ', f1_src)

print('   ')

print('accuracy_new: ', accuracy_new)
print('precision_new: ', precision_new)
print('recall_new: ', recall_new)
print('f1_new: ', f1_new)
```

Список источников

1. Audigier V., Niang N., Resche-Rigon M. Clustering with missing data: which imputation model for which cluster analysis method? 2021. — URL: https://arxiv.org/pdf/2106.04424. pdf (дата обращения 15.01.2022).
2. Carreras G., Miccinesi G., Wilcock A. [et al.]. Missing not at random in end of life care studies: multiple imputation and sensitivity analysis on data from the ACTION study // BMC Medical Research Methodology. 2021. Vol. 21 (1). P. 13. — DOI 10.1186/s12874-020-01180-y.
3. Demidenko E. Advanced Statistics with Applications in R. John Wiley & Sons Inc. 2019. — DOI 10.1002/9781119449195, restricted access.
4. Diday E. Optimisation en classification automatique et reconnaissance des forms // R.A.I.R.O. 1972. Vol. 3. P. 61–69. — URL: https://www.rairo-ro.org/articles/ro/pdf/1972/03/ro197206V300611.pdf (дата обращения 30.01. 2024).

5. Diday E. Optimisation en classification automatique. Institut national de recherche en informatique et en automatique. 1979. — DOI 10.1002/9781119449195, restricted access.
6. Forina M., Casolino C., Lanteri S. Cluster analysis: Significance, empty space, clustering tendency, non-uniformity. I - Statistical tests on the significance of clusters // Annali di chimica. (2003. Vol. 93. P. 55–68. — URL: https://www.researchgate.net/publication/10843361_Cluster_analysis_Significance_empty_space_clustering_tendency_non-uniformity_I_-_Statistical_tests_on_the_significance_of_clusters (дата обращения 31.01.2024).
7. Giesen, J., Kühne L., Lucas P. Sclow Plots: Visualizing Empty Space // Computer Graphics Forum. 2017. Vol. 36. P. 145–155. — DOI 10.1111/cgf.13175, restricted access.
8. Heymans M.W., Twisk J.W.R. Handling missing data in clinical research // Journal of Clinical Epidemiology. 2022. Vol. 151. P. 185–188. — DOI 10.1016/j.jclinepi.2022.08.016.
9. Hua C., Li F., Zhang C. [et al.]. A Genetic XK-Means Algorithm with Empty Cluster Reassignment // Symmetry. 2019 Vol. 11. P. 744. — DOI 10.3390/sym11060744, open access (дата обращения 29.01. 2024).
10. McGee G., Weisskopf M.G., Kioumourtzoglou M.-A. [et al.]. Informatively empty clusters with application to multigenerational studies // Biostatistics. 2020. Vol. 21. P. 775–789. — DOI 10.1093/biostatistics/kxz005, open access (дата обращения 15.01. 2024).
11. Mirkin B. Clustering: A Data Recovery Approach. — Chapman & Hall, 2nd edition. 2016. — DOI 10.1201/9781420034912, open access дата обращения 10.01.2024).
12. Pakhira M. A Modified k-means Algorithm to Avoid Empty Clusters // International Journal of Recent Trends in Engineering. 2009. Vol. 1. P. 220–226. — URL: https://www.researchgate.net/publication/228414762_A_Modified_k-means_Algorithm_to_Avoid_Empty_Clusters, open access (дата обращения 10.01.2024).
13. Pereira R.C., Santos M., Rodrigues P. [et al.]. MNAR Imputation with Distributed Healthcare Data // Progress in Artificial Intelligence. EPIA 2019. Lecture Notes in Computer Science. Vol. 11805. P. 184–195. — DOI 10.1007/ 978-3-030-30244-3_16.
14. Piernik M., Morzy T. A study on using data clustering for feature extraction to improve the quality of classification. Knowledge and Information Systems. 2021. Vol. 63 (7). P. 1771–1805. — DOI 10.1007/s10115-021-01572-6.
15. Raschka S., Mirjalili V. Python Machine Learning. Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. Packt, 2rd edition. 2019. — URL: http://radio.eng.niigata-u.ac.jp/wp/wp-content/uploads/2020/06/python-machine-learning-2nd.pdf.
16. Raykov Y.P., Boukouvalas A., Baig F. [et al.]. What to Do When K-Means Clustering Fails: A Simple yet Principled Alternative Algorithm // PLoS ONE. 2016. Vol. 11 (9). P. e0162259. — DOI 10.1371/journal.pone.0162259.
17. Reddy D., Jana P.K. Initialization for K-means Clustering using Voronoi Diagram // Procedia Technology. 2012. Vol. 4. P. 395–400. — DOI 10.1016/j.protcy.2012.05.061, open access (дата обращения 14.01.2024).
18. Rubin D.B. Inference and missing data // Biometrika. 1976. Vol. 63 (3). P. 581–592. — DOI 10.1093/biomet/63.3.581.

19. Shvedov A.S. Probability theory and mathematical statistics. Intermediate level. — Moscow: Higher School of Economics Publishing House, 2016. — URL: https://publications. hse.ru/en/books/179945401, restricted access.
20. Sokal R.R. Review [reviewed work: Optimisation en classification automatique. e. diday] // Journal of the American Statistical Association. 1984. Vol. 79 (387). P. 741–741. — DOI 10.2307/2288450, restricted access.
21. Sokal R.R., Sneath P.H. Principles of numerical taxonomy. — W.H. Freeman and Company, 1963. — URL: https://archive.org/details/principlesofnume 0000unse/page/n7/mode/2up, limited access (дата обращения 18.03.2024).
22. Tavallali P., Tavallali P., Singhal, M. K-means tree: an optimal clustering tree for unsupervised learning // The Journal of Supercomputing. 2021. Vol. 77. P. 5239–5266. — DOI 10.1007/s11227-020-03436-2, restricted access.
23. Xu D., Tian Y. A Comprehensive Survey of Clustering Algorithms // Annals of Data Science. 2015. Vol. 2. P. 165–193. — DOI 10.1007/s40745-015-0040-1, open access (дата обращения 30.01.2024).
24. Yadav A., Dhingra S. An Enhanced K-Means Clustering Algorithm to Remove Empty Clusters // International Journal of Engineering Development and Research. 2016. Vol. 4. P. 901–907. — URL: https://www.ijedr.org/papers/ IJEDR1604137.pdf, open access (дата обращения 30.01.2024).

Сведения об авторах

Пеникас Генрих Иозович, доктор экономических наук, профессор, руководитель проекта, Департамент исследований и прогнозирования, Банк России. 107016, г Россия, Москва, ул. Неглинная, д. 12. ORCID: 0000-0003-2274-189X. E-mail: *penikas@gmail.com*.

Henry I. Penikas, Doctor of Economics, Professor, Project Manager of the Department of Research and Forecasting, Bank of Russia. 12 Neglinnaya str., Moscow, Russia, 107016. ORCID: 0000-0003-2274-189X. E-mail: *penikas@gmail.com.*

Феста Юрий Юрьевич, менеджер Департамента противодействия мошенничеству, Сбербанк, 117312, Россия, Москва, ул. Вавилова, д. 19. ORCID: 0000-0001-9268-8284. E-mail: *festa.y.yura@gmail.com*.

Yury Yu. Festa, Manager of the Anti-Fraud Department, Sberbank. 19 Vavilova str., Moscow, Russia, 117312. ORCID: 0000-0003-2274-189X. E-mail: *penikas@gmail.com*.

Адрес сайта в сети Интернет: http://jem.dvfu.ru