Available online at www.sciencedirect.com
SCIENCE DIRECT®

Journal of Visual Languages and Computing
15 (2004) 1–35

Journal of Visual Languages & Computing

www.elsevier.com/locate/jvlc

Eureka!: an interactive and visual knowledge discovery tool

Giuseppe Manco[a,*], Clara Pizzuti[a], Domenico Talia[b]

[a] *ICAR-CNR, Via P. Bucci, 41C, 87036 Rende (CS), Italy*
[b] *DEIS, Università della Calabria, Via Bucci, 41C, 87036 Rende (CS), Italy*

Received 1 August 2002; received in revised form 7 March 2003; accepted 9 June 2003

Abstract

Visualization techniques may guide the data mining process since they provide effective support for data partitioning and visual inspection of results, especially when high dimensional data sets are considered. In this paper we describe *Eureka*!, an interactive, visual knowledge discovery tool for analyzing high dimensional numerical data sets. The tool combines a visual clustering method, to hypothesize meaningful structures in the data, and a classification machine learning algorithm, to validate the hypothesized structures. A two-dimensional representation of the available data allows users to partition the search space by choosing shape or density according to criteria they deem optimal. A partition can be composed by regions populated according to some arbitrary form, not necessarily spherical. The accuracy of clustering results can be validated by using different techniques (e.g. a decision tree classifier) included in the mining tool.
© 2003 Elsevier Ltd. All rights reserved.

Keywords: Clustering; Singular value decomposition; Visual data mining

1. Introduction

1.1. Motivations

The production of high dimensional data sets in different application domains has grown the interest in identifying new patterns in data that might be of value for the

*Corresponding author. Tel.: +39-0984-831728; fax: +39-0984-839054.
E-mail address: giuseppe.manco@icar.cnr.it (G. Manco).

1045-926X/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2003.06.001

holder of such data sets. Knowledge discovery is the process of analyzing data sets to identify interesting, useful and new patterns and trends in data [1]. The knowledge discovery process is a complex task that can involve the use of different data mining techniques. Data mining finds patterns or models that provide summarization of data while losing the least amount of information. Examples of models comprise clusters, rules, tree structures, and others. The combination of different models in a knowledge discovery process may help users in finding what is interesting and significant in large data sets.

Knowledge discovery is often referred as an interactive and iterative process that involves the following main phases: (1) data preparation and cleaning, (2) hypothesis generation, (3) interpretation and analysis. The hypothesis generation phase, generally, is completely automatic and realized using data mining algorithms based on machine learning and statistics techniques. A different approach aims at exploiting the perceptual and cognitive human abilities to detect the structure of data when a visual representation of data is available.

Visual data mining aims at integrating the human in the data exploration process, harnessing his interpretation abilities to large data sets. The basic idea of visual data mining is to present the data in some visual form, allowing the human to get insight into the data, draw conclusions, and directly interact in the data partitioning process [2]. Visual data mining is especially useful when little is known about data and the exploration goals are vague. Since the user is directly involved in the exploration process, shifting and adjusting the exploration goals is automatically done if necessary. Visual data mining exploits data visualization to guide the human user in the recognition of patterns and trends hidden in the data.

When high dimensional data sets are to be mined, visual data mining tools may benefit of the use of dimension reduction techniques that maintain the main features of data. A reliable data mining system must provide estimate accuracy or error of the extracted model (knowledge) of the mining process. This accuracy estimation can compensate for the deficiency that imprecise analysis of data visualization may produce.

1.2. Related work

The integration between information visualization and data mining techniques is an outstanding research field that received a lot of attention in the last few years. Originally, visualization in the knowledge discovery process was mainly used in the presentation of the results to provide a better understanding of the information discovered. The need to tightly combine automatic data mining algorithms and visualization technology to improve the quality and speed of the visual data mining process now is largely recognized. For example, Keim in [3] argues on the importance of realizing such an integration and proposes a classification of visual data mining techniques based on the data type to be visualized, the visualization technique and the interaction technique. In [4] the most significant experiences coming from data mining and data visualization research fields are reported. Several visualization techniques have been developed to support the various steps of the knowledge discovery process. In particular, some proposals to assist data preprocessing and exploration [5–8], clustering [9–12], classification [13,14] and associative rule generation [15] have been done. In the following the main proposals for clustering and classification are described.

In [9] a graphical tool, named *HD-Eye*, allows the user to choose clusters in lower dimensional spaces. *HD-Eye* is built as an extension of the *OptiGrid* [16] algorithm for clustering high dimensional data sets. *OptiGrid* finds clusters by partitioning the data set. To this end it uses a multidimensional grid defined by a set of separators having a dimensionality lower than the original space. Separators are chosen in regions having the minimal point density. Choosing the best separators and determining the projections of the search space is difficult thus, instead of doing these tasks automatically, like *OptiGrid*, *HD-Eye* employs a visualization technique that allows the user to deeply understand the data set and enables him to identify projection and separators straightforwardly. When the dimensionality increases, however, the task of the user becomes more difficult.

OPTICS, proposed in [17], creates a one-dimensional ordering of the database representing its density-based clustering structure. In such a one-dimensional ordering, points within a cluster are close, in an ideal density-based clustering structure. The algorithm works for arbitrary-sized datasets, and does not explicitly produce a clustering of the dataset, as its main purpose is to serve as a basis for automatic and interactive cluster analysis. To this end, the cluster-ordering can be represented graphically by means of appropriate visualization techniques, particularly suitable for interactive exploration of the intrinsic clustering structure.

IPCLUS, proposed by Aggarwal in [10], is a system that, on the basis of a user-defined support consisting of the minimum fraction of points of the data set that a cluster must contain to be considered statistically significant, suggests the best subspaces in which clusters can be identified. The user then separates points to form clusters and this process is repeated until almost all the points have been assigned to a cluster. The projections in which it is possible to clearly distinguish groups of points are called *well polarized*. Polarizations are determined by considering a set of k records from the data set, called *polarization anchors*, and a subspace is identified if data is clustered around these anchors. Repeatedly sampling the data set for searching anchors, allows to find the best subspaces.

Kandogan in [11] uses the *Star Coordinates* visualization technique to discover hierarchical clusters. In Star Coordinates, coordinate axes are arranged on a two dimensional surface, each axis sharing the same origin. The minimum data value is mapped to the origin and the maximum value is mapped to the other end of the coordinate axis. Unit vectors of each coordinate are obtained by scaling the data values to the length of the coordinate axes. The location of each point is determined by considering the sum vector of its values on each dimension. The Star Coordinate visualization system allows to better understand the data set by means of a number of interaction features. Such features are scaling, changing the length of an axis, rotation, modifying the direction of the unit vector of an axis, marking, selection of a point or all points in a rectangle, range selection, selection of data value ranges, and many others. By using all these characteristics, the user can dynamically interact with the system to discover outliers, trends, and clusters easily.

Poulet [12] presents a system called *CIAD*+ that combines automatic algorithms, interactive algorithms and visualization tools. CIAD+ allows for the construction of an interactive decision tree so that the user is helped by an automatic algorithm based on the Support Vector Machine [18] to compute the best split of an attribute. This algorithm is then modified to allow for clustering by using the *k*-means automatic algorithm to support the user. Furthermore a visualization algorithm is used to explain the results obtained.

A collaborative technique that helps the user in the task of data classification for decision tree construction is described in [13]. The system, called *Perception-Based Classification* (*PCB*), visualizes the training data with their class label and maps classes to different colors. The interaction with the system consists of the data interaction steps and the knowledge interaction steps. The data interaction steps allow for data visualization and the possibility of selecting an attribute and its split points. The knowledge interaction steps present the decision tree built so far and enable the user to assign a class label to a node, visualize any node and backtrack on the decisions already made. *PCB* has been extended in [14] to support also categorical attributes, a more sophisticated visualization technique and different possibilities of collaboration between the user and the system. The user can decide to build the parts of the decision tree by combining both manual and automatic construction.

All the approaches described aim at facilitating the work of the human user in a particular phase of the knowledge discovery process. To our knowledge, poor support has been given to the overall visual interactive modelling of the KDD process: among the few proposals, we mention the CRISP-DM process model and the Predictive Modelling Markup Language proposal.[1] A cooperation approach that involves visualization and mining techniques such that results obtained by one technique may feed another one would result in a more powerful and synergistic approach to data exploration and discovery. The paper intends to give a contribution in this setting.

1.3. Contribution

In this paper we describe a human assisted knowledge discovery tool, named *Eureka*! that combines a visual clustering method used to hypothesize meaningful structures in the data and a validation methodology used for assessing the quality of the hypothesized structures. Examples of validation techniques that can be exploited are based on *external* criteria, when predefined structures are known, and *internal* criteria, when a classification machine learning algorithm is adopted. The tool uses the optimal dimensionality reduction method, known as *Singular Value Decomposition* (*SVD*) [19], to obtain a two-dimensional representation of the available data, and iteratively asks the user to suggest a suitable partition of such a representation.

[1] Details can be found at `http://www.crisp-dm.org` and `http://www.dmg.org`.

The choice of a partition is requested to the users; they can identify clusters of any shape or any density. In fact, a partition could provide a separation of dense regions from regions containing sparse data, or it could be composed of arbitrary polygonal or spherical regions. The accuracy of the clustering results can be validated by using a decision tree classifier included in the mining tool.

The main contribution of the paper is the combination of clustering and classification data mining techniques with an interactive visual approach in a tool for knowledge discovery. In addition, a structured methodology for interactively developing a Knowledge discovery process in all its stages has been developed. The methodology is based on a fixed model of interaction in which the various phases of the discovery process (data preparation, data mining and model discovery, and model evaluation) interact by means of a structured interface.

Eureka! has been implemented [20] mainly as an extension of the Weka machine learning library [21]. Weka is a Java library that defines standard interfaces for data sets loading and preprocessing (e.g., filter definition), mining algorithms and results representation. The integration of the *Eureka*! functionalities in the Weka environment results in a rather versatile mining environment, in which most knowledge discovery techniques can be used cooperatively.

1.4. Plan of the paper

The rest of the paper is organized as follows. Section 2 provides a brief introduction of the mathematical technique underlying the clustering tool and describes the rationale of the proposed technique. In Section 3 we describe the interaction metaphor implemented into the system. In particular, Section 3.2 covers the cluster generation technique, while Section 3.3 is concerned with the cluster validation technique. Section 4 illustrates the results of the proposed technique on two particularly significant datasets. Finally, Section 5 outlines the main advantages of the proposed approach and adds some references to future extensions of the implemented system.

2. Clustering via singular value decomposition

SVD is a powerful technique in matrix computation and analysis that has been introduced by Beltrami in 1873 [22]. More recently it has been used in several applications such as solving systems of linear equations, linear regression [19], pattern recognition [23], statistical analysis [24], data compression [25], matrix approximation [26] and information retrieval [27].

A *singular value decomposition* of an $n \times m$ matrix X is any factorization of the form

$$X = U \times \Lambda \times V^{\mathrm{T}}$$

where U is an $n \times n$ orthogonal matrix, V is an $m \times m$ orthogonal matrix and Λ is an $n \times m$ diagonal matrix with $\lambda_{ij} = 0$ if $i \neq j$. It has been shown that there exist matrices

U and V such that the diagonal elements of Λ are sorted: $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_m$. The diagonal elements λ_i are called *singular values* of X and it has been shown that they are the square root of the eigenvalues of the matrix $X^{\mathrm{T}}X$.

The decomposition can equivalently be written as

$$X = \lambda_1 u_1 \times v_1^t + \lambda_2 u_2 \times v_2^t + \cdots + \lambda_m u_m \times v_m^t$$

where u_i and v_i are column vectors of the matrices U and V respectively, λ_i are the diagonal elements of Λ, and it is known as *spectral decomposition* [24]. *SVD* reveals an important information about the rank of the matrix X. In fact, if $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_r \geqslant \lambda_{r+1} = \cdots = \lambda_m = 0$ then r is the rank of X [28].

Geometrically this factorization defines a rotation of the axis of the vector space defined by X where V gives the directions, Λ the strengths of the dimensions, and $U \times \Lambda$ the position of the points along the new axis.

Intuitively, the U matrix can be viewed as a similarity matrix among the rows of X, i.e., the objects of the data set, the V matrix as a similarity matrix among the columns of X, that is the features that describe an object, and the Λ matrix gives a measure of how much the data distribution is kept in the new space [29].

In the data mining area, *SVD* can be used to identify *clusters* by analyzing the U matrix. *Clustering* is a data mining task [1] for unsupervised classification that consists in partitioning large sets of data objects into homogeneous groups [23,29–31] when no predefined knowledge is available. Clusters can be described as regions of the search space containing points which are close to each other.

By visualizing the $U \times \Lambda$ matrix and considering only the first d dimensions, where $d \leqslant 3$, we obtain a compressed representation of the X matrix that approximates it at the best. In particular, if

$$\sum_{i=1}^{d} \lambda_i \Big/ \sum_{i=1}^{m} \lambda_i \geqslant 0.95$$

then the original pattern matrix X is well approximated by the first d terms of the spectral decomposition [29]. The d kept terms are known as the *principal components* [24].

Fig. 1 shows the advantages of the SVD transformation for clustering data. The image on the left is a two-dimensional visualization of a dataset X containing 19 dimensions (in which only the first two dimensions are exploited). As we can see, no suitable separation can be recognized. The image on the right is a two-dimensional representation of the matrix $U \times \Lambda$, built considering only the first two components. In this case, the data exhibits neat separations in the new space.

The above observations suggest an interactive methodology for building a cluster tree, i.e. a tree in which:

- Each node represents a subset of the rows of the original data matrix X;
- The root node represents X;
- If $X_{i_1}, \ldots, X_{i_k}$ are the sons of a given node X_i in the cluster tree, then $\bigcup_j X_{i_j} = X_i$ and $X_{i_h} \cap X_{i_k} = \emptyset$;
- leaf nodes represent homogeneous groups.

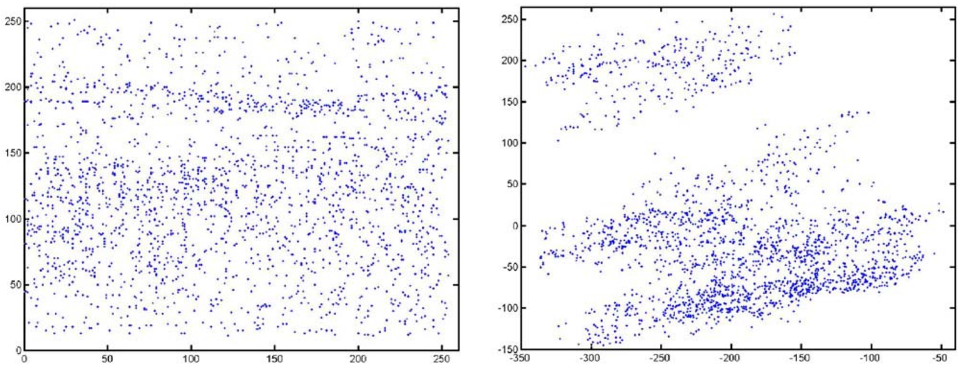


Fig. 1. SVD decomposition for clustering.

```
Input : a data matrix $X$;
Output : a partition $C = \{X_1, \ldots, X_h\}$ of $X$ in $h$ clusters.
Method :
    • Set $X$ as the root node of the cluster tree.
    • Repeat
        1. Select a leaf node $X_i$ from the cluster tree.
        2. Apply the SVD decomposition $X_i = U \times \Lambda \times V^T$.
        3. Provide a two-dimensional visualization of $U \times \Lambda$ according to
           the two most significant components.
        4. If a suitable partition $X_{i_1}, \ldots, X_{i_k}$ is detected
            – Add each $X_{i_j}$ to the cluster tree as a son of $X_i$.
      until no suitable partition is found.
    • Return the leaf nodes of the computed cluster tree.
```

Fig. 2. Interactive SVD-based clustering algorithm.

The general schema of the algorithm is sketched in Fig. 2. Initially, a root node (corresponding to the entire dataset) is provided. Next, the algorithm is recursively applied to all the available leaf nodes of the tree. In analyzing a node, an SVD decomposition is computed on the submatrix associated with the node, and a two-dimensional visualization is provided. The default choice is based on the selection of the two most significant components. However, users may select two arbitrary components as needed. Hence, if a suitable partition is detected, new nodes are generated. Each generated node corresponds to one of the detected partitions. The algorithm terminates when no new nodes are generated (that is, when no eligible partitions are detected from the currently available leaf nodes).

3. *Eureka*!: a tool for interactive knowledge discovery

In this section we present the main features of *Eureka*!. A general overview of the system that describes the interaction paradigm implemented in *Eureka*! is provided. Next, we describe the main features of the system by means of a well-known example: the *image segmentation* database (referred as *segment* in the following). The data set was taken from the UCI Machine Learning Repository [40] and describes a set of instances drawn randomly from a database of 7 outdoor images. The images where hand-segmented to create a classification for every pixel. The data set consists of 19 numeric attributes, describing the features of a 3×3 region that the instance represents.

3.1. An overview of the system

Eureka! is a semiautomatic tool for interactive visual knowledge discovery that integrates a visual clustering method, based on the Singular Value Decomposition technique, and a methodology to validate the clustering results. *Eureka*! has been designed and implemented with the aim of making repeatable the knowledge discovery process on a data set and storing the steps done during the overall process into a repository in order to use it again at a later time. Thus *Eureka*! implements a fixed model of interaction with users, in which the various steps of the data mining process are represented in a uniform way and executed according to a predefined schema. To this end *Eureka*! generates a hierarchical structure that describes the overall Knowledge Discovery in Databases (KDD) process called *Repository*. The schema that models the KDD process is shown in Fig. 3.

Intuitively, an analysis addressing some predefined objectives defines a *business-process*. A given business process is composed by one or more *kdd-process* items. Such items have the main objective of describing the meta-schema of each possible instantiation of the analysis: in particular, a *kdd-process* uses a given data set (with a given structure described by the *input-schema* item), and it is subject to a given number of *preprocessing* steps, thus providing a *preprocessed-schema* item. The *dm-process* module describes the adopted data mining techniques, as well as the parameters to correctly apply such techniques. Finally, a *kdd-instance* contains one or more possible instantiations of a KDD process. In particular, it contains an input data set conforming to the *input-schema*, the data set resulting from the preprocessing steps described in *preprocessing*, and the resulting patterns obtained from the application of the data mining algorithms.

An interesting aspect of the above described model is the capability of allowing the interaction between different analysis. An input schema of a *kdd-process* instance can be either a table or the output of a different *kdd-process* instance. For example, the result of a clustering process, in which each instance is labelled by a cluster number,

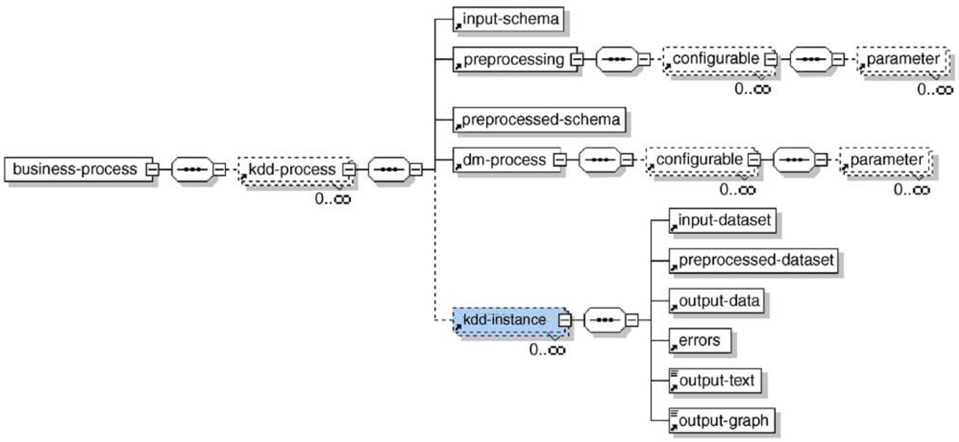
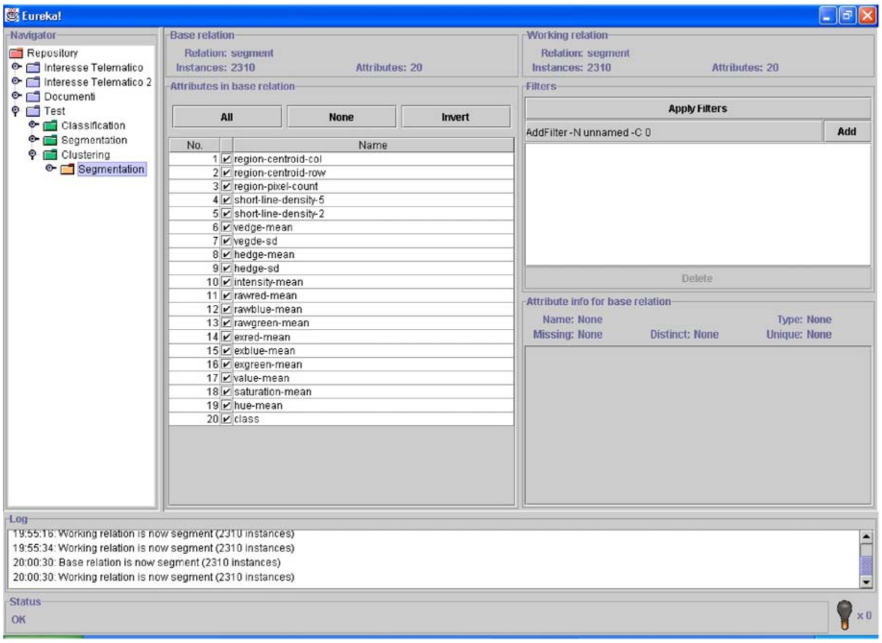


Fig. 3. The interaction model implemented in *Eureka*!.



Base relation

Relation: segment
Instances: 2310 Attributes: 20

Attributes in base relation

No.		Name
1	✓	region-centroid-col
2	✓	region-centroid-row
3	✓	region-pixel-count
4	✓	short-line-density-5
5	✓	short-line-density-2
6	✓	vedge-mean
7	✓	vegde-sd
8	✓	hedge-mean
9	✓	hedge-sd
10	✓	intensity-mean
11	✓	rawred-mean
12	✓	rawblue-mean
13	✓	rawgreen-mean
14	✓	exred-mean
15	✓	exblue-mean
16	✓	exgreen-mean
17	✓	value-mean
18	✓	saturation-mean
19	✓	hue-mean
20	✓	class

Working relation

Relation: segment
Instances: 2310 Attributes: 20

Filters

AddFilter -N unnamed -C 0

Attribute info for base relation

Name: None Type: None
Missing: None Distinct: None Unique: None

Log

```
19:55:16: Working relation is now segment (2310 instances)
19:55:34: Working relation is now segment (2310 instances)
20:00:30: Base relation is now segment (2310 instances)
20:00:30: Working relation is now segment (2310 instances)
```

Status

OK

x 0

Fig. 4. *Eureka*! interface. The left part contains the Navigator area, in which the Repository is managed.

can be set as input for a classification process, in which a predictor system for the labelling attribute is built.

Fig. 4 shows the graphical interface of *Eureka*!. It is composed of three main areas. On the left, the component referred as *Navigator* allows the generation and navigation of the *Repository*. The *Repository* is a hierarchical tree structure that maintains the step sequence done during the overall KDD process. It thus allows the creation and updating of a *business-process*. The bottom part of the interface provides messages about the *kdd-process* execution and the right part shows the current running task. A running task can be represented by any of the (extended) Weka functionalities. By suitably managing the *Repository*, users can exploit the interaction among such functionalities.

The methodology used in *Eureka*! is shown in Fig. 5. An input data set is transformed into a reduced data set by applying the SVD algorithm and visualized with respect to any two principal components. The user is then asked to choose a portion of the search space he deems interesting. The selected portion is identified as a cluster and the process is repeated on the remaining data until the user judges satisfactory the grouping obtained. At this point each tuple of the data set is labelled with the corresponding class detected by the user and a validation technique can be applied to verify the accuracy of the model found, i.e. its correspondence to the true homogeneous subgroups in the data. Thus, if the correspondence is low, the user can backtrack on his choices and provide an alternative division of the search space, otherwise he can save the process done, and its results. A suitable such validation technique can be the exploitation of a decision tree inducer over the class label, in which low misclassification errors should substantiate the detected groups, or the exploitation of prior knowledge about the separability of the given dataset.

From the discussion above, we can detect two main steps in the clustering process: cluster generation and visualization, and cluster interpretation and validation. Let us analyze them in deeper details.

3.2. Cluster generation and visualization

As mentioned before, *Eureka*! implements an interactive divisive hierarchical clustering algorithm such that at each step clusters can be chosen visually in a two-dimensional space. Such an approach has the advantage of allowing to choose clusters that do not necessarily obey to predefined structural properties, such as

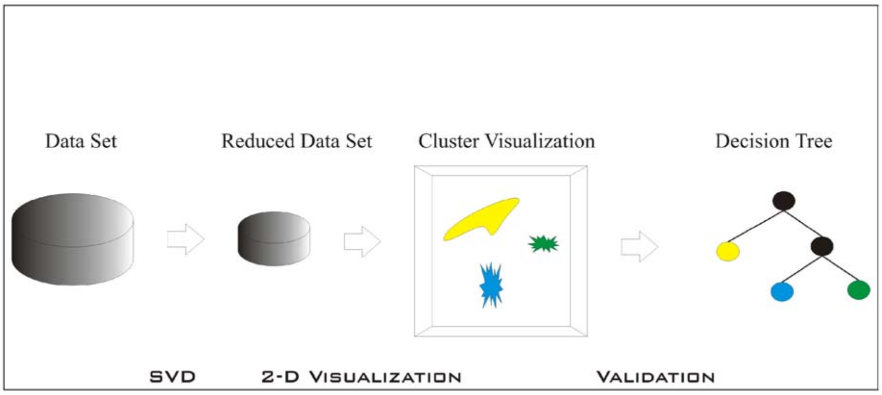


Fig. 5. Main steps of the data mining methodology.

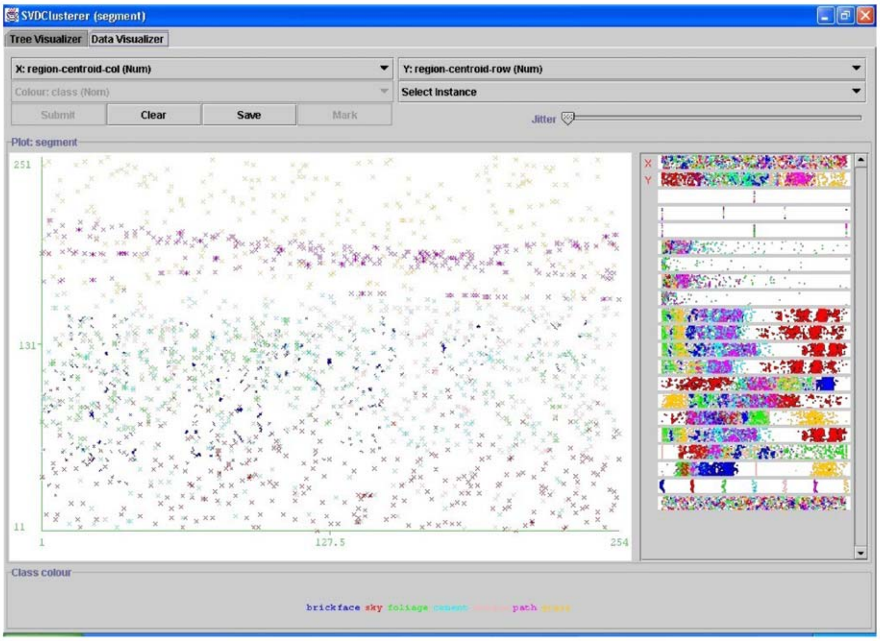


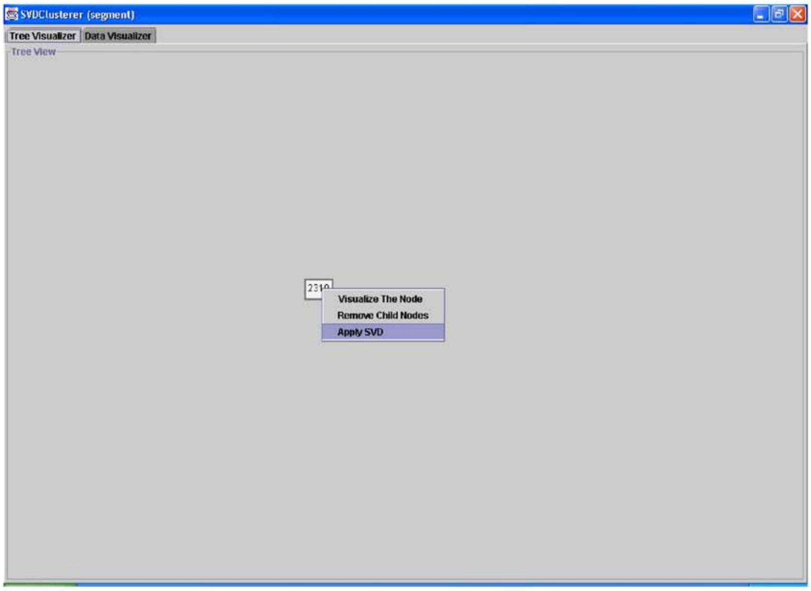
Fig. 6. Visualization Pane. In the rightmost part, the whole set of possible dimensions is shown. Each dimension has associated a small graphic providing a summary of the related data distribution. By selecting as coordinates (represented by the labels X and Y in the pane) any two dimensions, a graphical representation of the dataset is provided in the main area of the pane, according to the selected coordinates.

density or shape [32,33]. At each step, users can choose the cluster according to the criteria that are more likely to be applied, according to the domain knowledge at hand. As a final result, the clustering task provides a representation of the available dataset in a cluster tree, in which each leaf node of the tree represents a suitable partition of the original dataset.

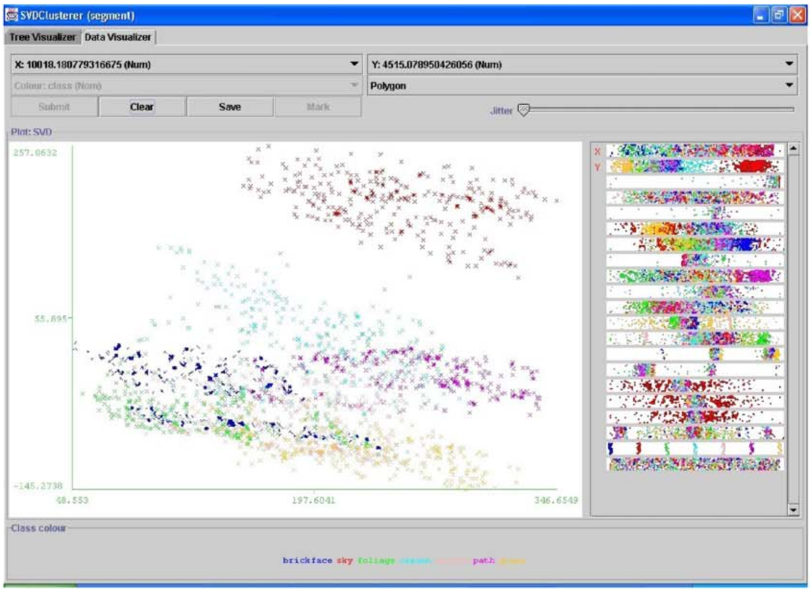
To this end, after the data set has been selected and preprocessed, the clustering task starts with a cluster tree containing a single node. The node represents the dataset, and has associated a two-dimensional visualization in the corresponding Data Visualization pane, as shown in Fig. 6. In such a visualization, each point corresponds to an instance in the given dataset.[2]

The clustering process starts by choosing to apply the SVD transformation to the data set, as shown in Fig. 7(i). The corresponding Data Visualizer pane, shown in Fig. 7(ii), represents the $U \times \Lambda$ matrix corresponding to the selected node. The rows

[2]For ease of presentation, in the following figures we associate with each instance in the two-dimensional representation a color, representing the true class in the *segment* database. The class information associated with each instance can give the reader the intuition behind the accuracy of the partitioning process during the cluster generation.



(i)



X: 10018.180779316675 (Num)
Y: 4515.078950426056 (Num)
Colour: class (Nom)
Polygon

(ii)

Fig. 7. Visual interaction in the cluster generation task. Initially, the cluster tree contains a single node, representing the entire dataset. (i) *Eureka*! runs the SVD transformation to the overall data set. (ii) Visualization of the transformed data set with respect to the two principal components.

on the right part of the pane represent the dimensions, sorted according to decreasing values of the corresponding eigenvalues λ_i. Since the representation can only be two-dimensional, users have to choose which dimensions to exploit as X and Y coordinates. By default, the system chooses the first two dimensions, corresponding to the highest eigenvalues in the matrix Λ.

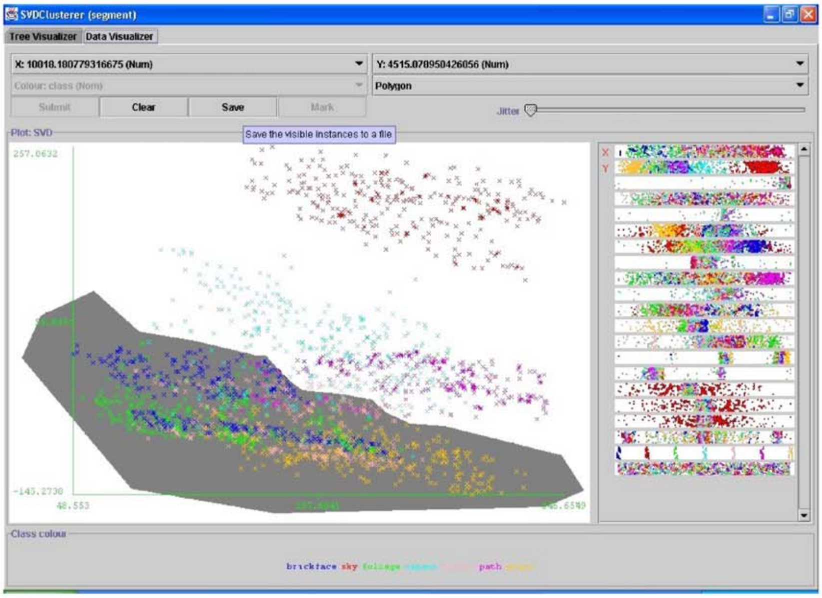
By visualizing the transformed data set in Fig. 7(ii), we can clearly distinguish at least three separate regions. In our visualization, separate regions represent clusters, i.e., elements that can be grouped together. In order to identify clusters, we need to draw the borders of a given region. More precisely, we can choose a region, and separate it from the rest of the space that is represented. *Eureka*! allows users to separate a region by choosing the most appropriate shape (e.g. a rectangle or a polygon), as shown in Fig. 8(i). Once a region has been selected, we can store such a selection, thus obtaining a partition of the original space in a cluster tree representing two different groups, as shown in Fig. 8(ii). The right node represents the selected region, and the remaining items (points in the space) are represented by the left node. Notice that now the root node contains information about the splitting criteria adopted: in particular, the numerical values represent the eigenvalues corresponding to the dimensions that guided the split. As we shall see, information about the splitting dimensions are associated with each node actually split.

We can choose any node in the Tree Visualizer Pane, thus enabling the corresponding visualization in the Data Visualizer Pane. In Figs. 9(i) and (ii), the 1358 points of the left node and the 952 points of the right node are visualized. New nodes can then be recursively split. In particular, the right node shows a clear separation among two different regions, so is worth a further splitting. This is shown in Figs. 10(i) and (ii).

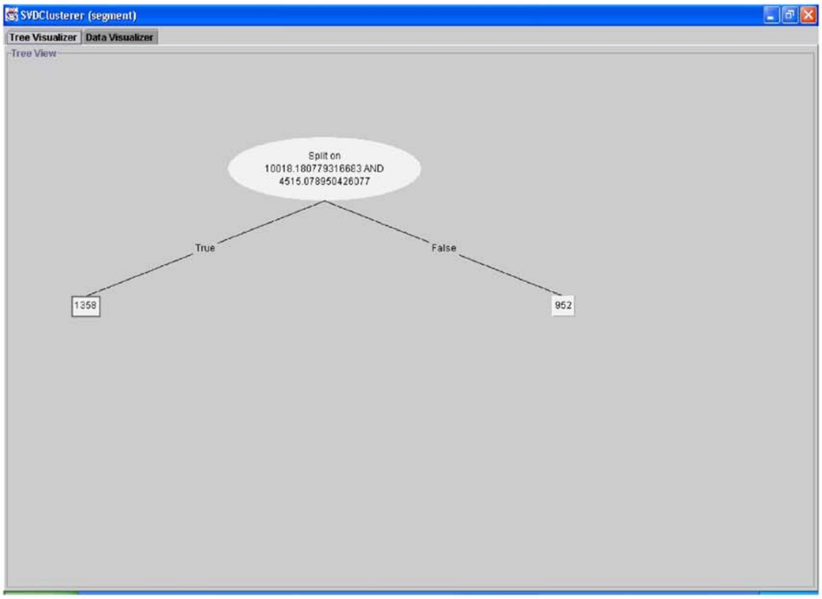
An interesting aspect of the tool is the capability of changing axes in the two dimensional representation. By default the mining tool provides a visualization of the first two dimensions (corresponding to the highest eigenvalues of the matrix Λ). However, by clicking over a given dimension among those shown in the left part of the Data Visualization pane, users can change such a visualization as needed. A different visualization of the node shown in Fig. 9(i) and reproduced in Fig. 11(i), can be obtained by representing the Y axis using the fourth dimension in the SVD representation (Fig. 11(ii)). Different visualizations can help users in the cluster identification process.

An unsatisfactory partition can be removed by directly acting on the cluster tree. For example, if the analysis of a node does not put in evidence a clear separation of the regions in the given data set partition, we can choose to delete it, as shown in Fig. 12(ii). Many further choices are available, in order to make separation as accurate as possible. In particular, we can choose non-convex regions, as shown in Fig. 8(i), or we can choose whether to locally apply the SVD decomposition to a given node (Fig. 12(i)) or not.

The interaction of users within the cluster tree is stopped when no further significant splits can be detected. In the *segment* example, we obtained a tree, represented in Fig. 16(i) containing 29 nodes and 17 leaves. Each internal node of

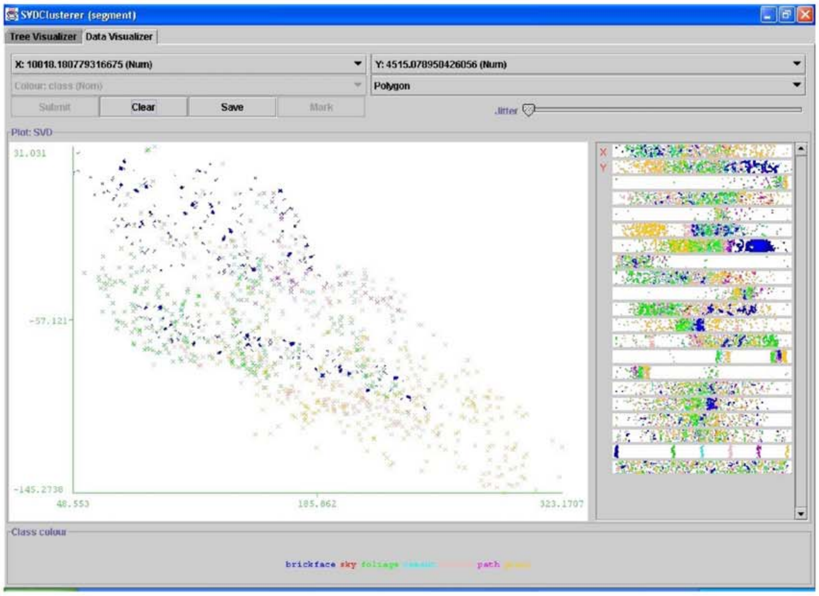


(i)

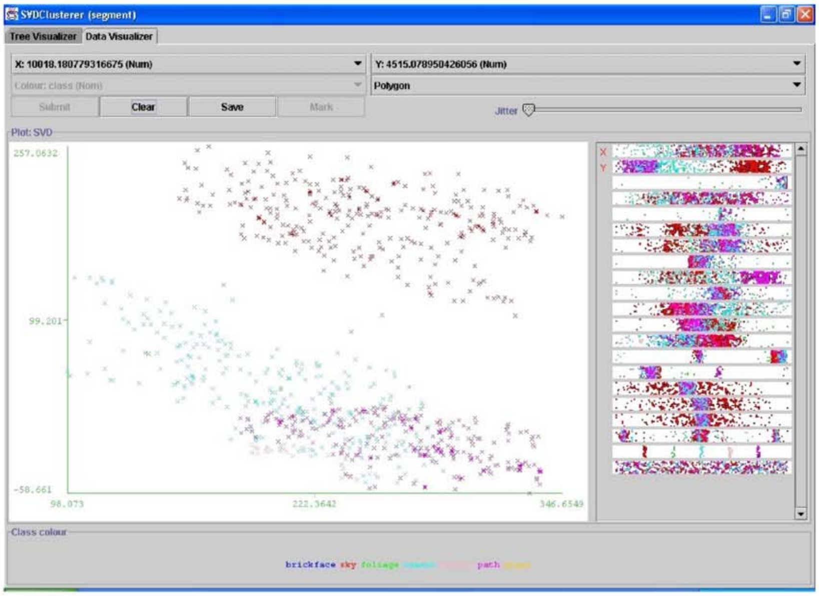


(ii)

Fig. 8. Visual interaction in the cluster generation task. (i) A region (corresponding to a portion of the dataset) is selected in the Visualization pane. By submitting the selection, the dataset is split in two main subsets. (ii) Cluster tree after the split.

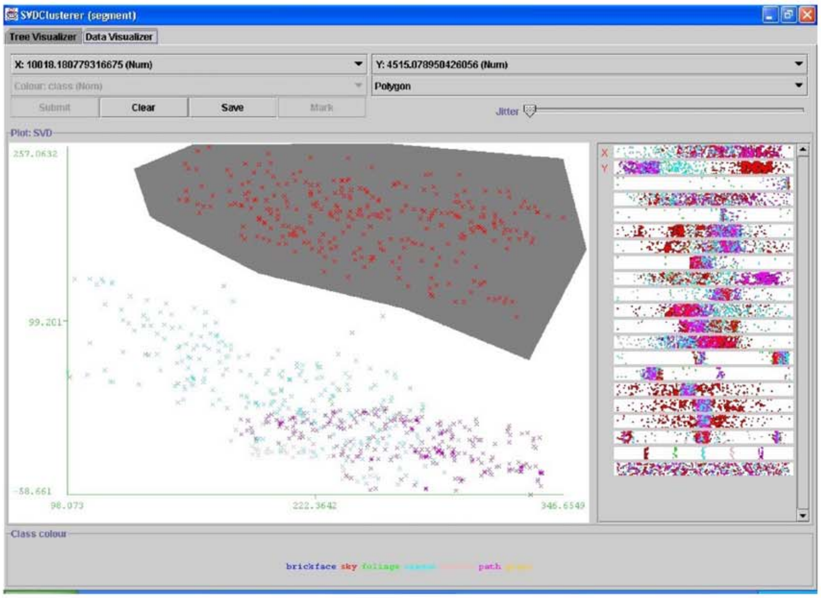


(i)

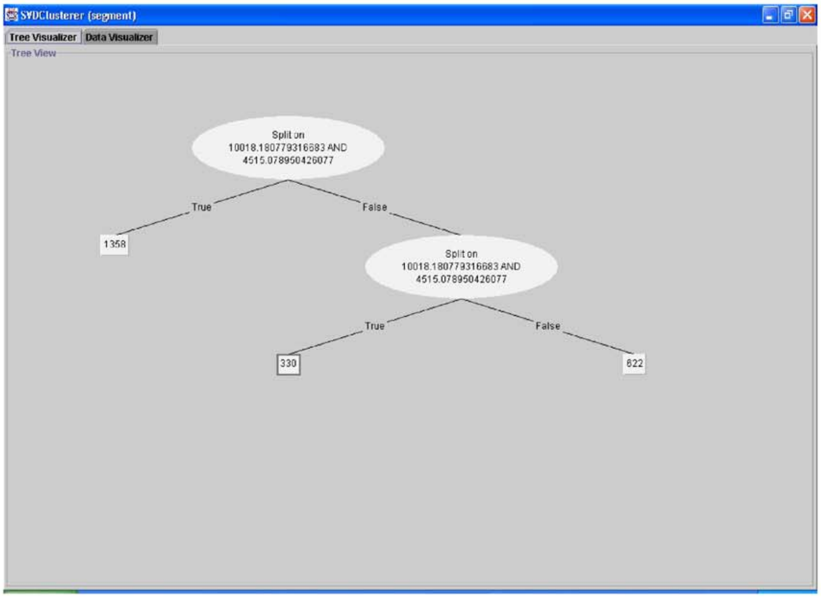


(ii)

Fig. 9. Visual interaction in the cluster generation task. (i) Visualization of the selected region (corresponding to the leftmost node in the cluster tree). (ii) Visualization of the remaining region (corresponding to the rightmost node in the cluster tree).

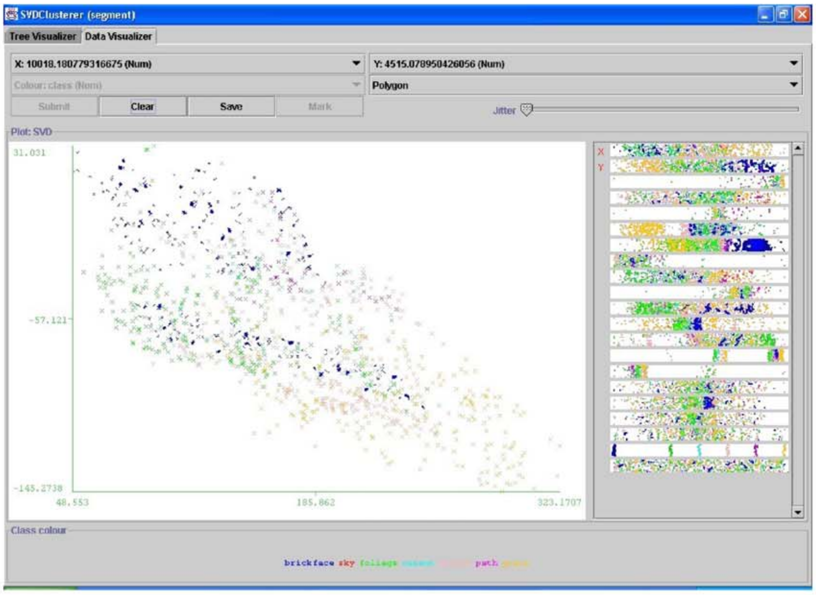


(i)



(ii)

Fig. 10. The visual interaction can be recursively applied to each subnode of the cluster tree, by visually analyzing such nodes and inspecting them from different perspectives. (i) The rightmost node in the cluster tree of Fig. 8(i) is analyzed, and (ii) a suitable splitting detected.

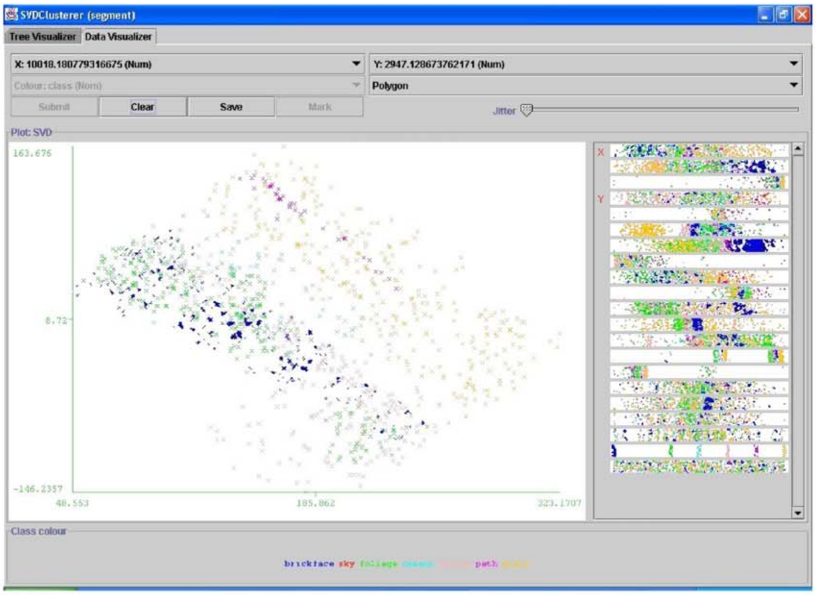


X: 10018.180779316675 (Num)

Y: 4515.078950426056 (Num)

Polygon

(i)



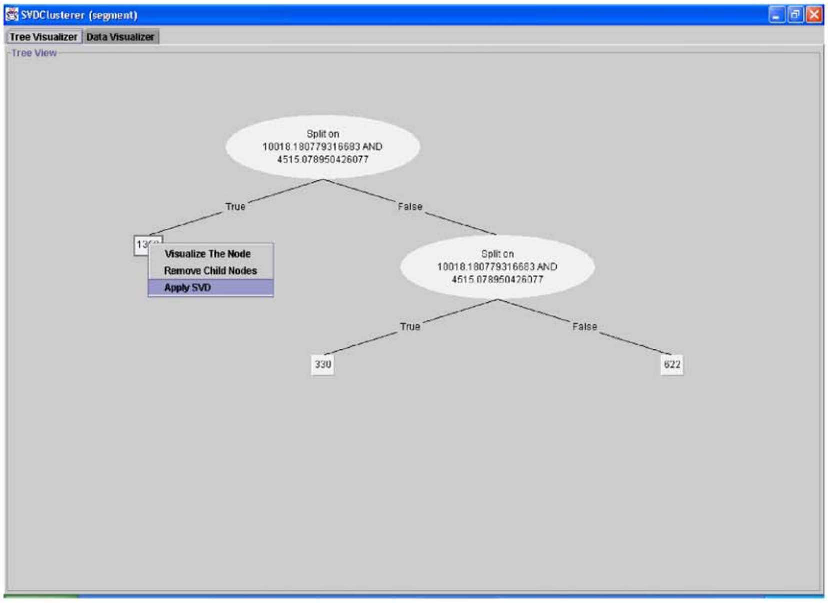
X: 10018.180779316675 (Num)

Y: 2947.128673762171 (Num)

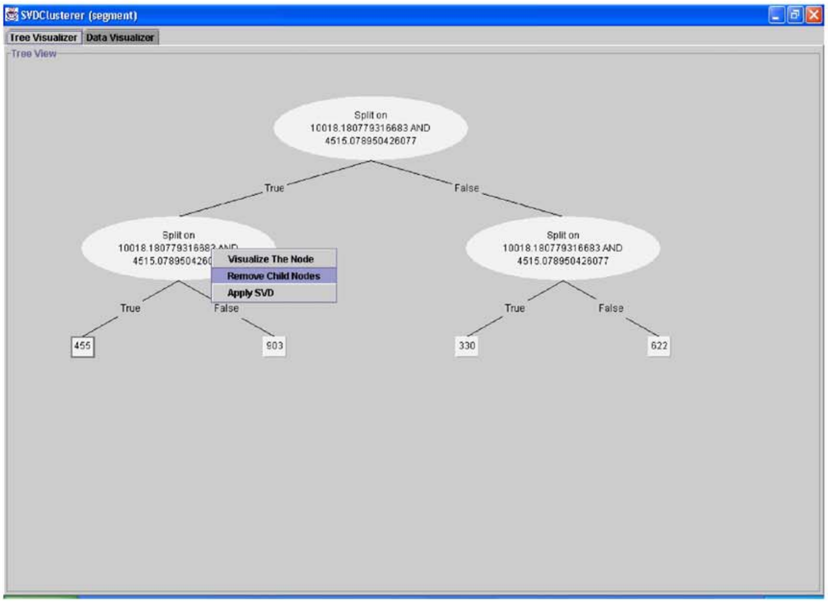
Polygon

(ii)

Fig. 11. (i) Axis modification: the leftmost node of Fig. 10(ii) visualized according to components 1 and 2, and (ii) according to components 1 and 4. In the second figure we can clearly distinguish two well-separated rectangular regions.



(i)



(ii)

Fig. 12. Visual interaction within the cluster tree: (i) To each node the SVD decomposition can be reapplied to the portion of the data set represented by the current node. (ii) Deletion of unsatisfactory partitions.

such a tree represents a split according to some chosen dimensions (represented by the corresponding eigenvalues in the Λ matrix in the visualization). The leaves represent a partition of the data set in 17 groups, as detailed in Fig. 16(ii).

To summarize, we can identify the following features to the *Eureka*! interaction:

1. Dynamic SVD decomposition of a given partition;
2. Interaction with the cluster tree;
3. two-dimensional visualization with interchangeability among the dimensions of the decomposition.

3.3. Cluster interpretation and validation

The above described methodology is a semi-supervised technique, which in principle may suffer of the two typical problems which usually arise when dealing with clustering:

- The assessment of the quality of the results. Since clustering algorithms define clusters that are not known a priori, irrespective of the clustering methods, the final partition of data requires some kind of evaluation [33]. Such an evaluation is useful to measure the adequacy of the discovered structure so that it can be interpreted objectively. The adequacy of a clustering structure refers to the sense in which the clustering structure provides true information about the data: a clustering structure is "valid" if it fits the dataset, i.e. if the discovered clusters correspond to the actual homogeneous groups in the dataset. Typically, the validity of a cluster tree has to be devised using appropriate criteria and techniques, which may be justified on the basis of the application area. When no homogeneous structure naturally holds in the data, such criteria should indicate that no suitable clustering scheme can explain the data.
- The interpretation of the clustering results. A suitable clustering scheme is characterized by some relevant features, which differentiate each component of the cluster structure. Such features are based on the properties of the data: for example, each cluster may be characterized by different ranges of values for the attributes of the data. It is clear that, in order to associate the appropriate meaning with each cluster in the cluster structure, the relevant features need to be detected. Usually, visual exploration techniques, together with descriptive statistics, can be well-suited for interpreting the results of a clustering: examples can be the visualization of the distribution of the data within each cluster, and the computation of the most frequent (combinations of) values which occur.

Both these issues contribute in devising a criterion for accepting or rejecting a clustering result. A clustering structure is acceptable if it is valid according to a given criteria, if it is unusual in some sense, and yields actionable (i.e. potentially useful) information. In the following we describe the methodologies employed in *Eureka*! to accomplish such tasks as above.

In [29,33,34], three main methods are described for assessing the validity of a clustering result:

- *external criteria*, when clustering results are evaluated according to a pre-specified structure.
- *internal criteria*, when clustering results are evaluated in terms of the quantities that are computable from the available data (e.g. similarity matrix).
- *relative criteria*, when evaluation takes place in comparison with other clustering schemes.

It is worth noticing here that the adoption of external criteria helps to understand the clustering results. Indeed, a pre-defined structure can be interpreted as the explanation of the corresponding data by means of some hypotheses. As a consequence, the correspondence of a cluster to one of such pre-defined structures implies the interpretation of such a cluster according to the corresponding hypothesis. The situation is different when internal or relative criteria are adopted. In these cases, the evaluation of the validity of the discovered clusters does not necessarily provide help on the interpretation of such clusters, so one has to resort to further techniques for interpreting the results of the clustering algorithm.

Eureka! implements both an external criterion, mainly based on the evaluation of the adequacy of the clustering result to prior knowledge about the separability of the given dataset, and an internal criterion, in which a quality index is devised for the clustering scheme under examination. If substructures can be hypothesized (e.g. if knowledge about the true classes is available) we can compute and analyze a *contingency table* describing the correspondence among the discovered structures and the hypothesized structures. A high degree of agreement can be interpreted as a high-quality clustering; on the other side, a low degree of correspondence requires the user to backtrack its choices in the division of the search space.

As far as internal criterion is concerned, a suitable validation technique can be the exploitation of a decision tree inducer over the class label. Low misclassification errors should substantiate the detected groups. Thus, if the misclassification error is high, the detected clustering scheme is judged inadequate, and is rejected; if, on the other side, the misclassification error is low, the discovered clustering scheme can be accepted. The exploitation of a decision tree inducer as a internal criteria is particularly significant, in that it can be devised as an effective and intuitive mean for interpreting clustering results.

3.3.1. Validation based on contingency tables

When a pre-specified structure is available, *Eureka*! can generate a contingency table m, in which each column represents a cluster c_j in a clustering scheme $\mathscr{C} = c_1, \dots, c_k$ (obtained in the cluster generation phase), and each row represents a partition p_i in an ideal partitioning $\mathscr{P} = p_1, \dots, p_s$. The term m_{ij} corresponds to the number of tuples in the original dataset that were associated with cluster c_j and actually belong to the ideal partition p_i. Such a contingency table is further exploited to associate with the clustering scheme two main classes of indexes.

- *local quality indexes* are useful to measure the quality of a single cluster. In particular, a local quality index measures the dispersion of the values within a row/column. Two main local quality indexes are computed in *Eureka*!:
  1. the *local precision* of a cluster c_j w.r.t. partition p_i, i.e. the number of instances belonging to both c_j and p_i, relative to c_j
  $$p_{ij} = \frac{|c_j \cap p_i|}{|c_j|}$$
  2. the *local recall* of a cluster c_j w.r.t. partition p_i, i.e. the number of instances belonging to both c_j and p_i, relative to p_i
  $$r_{ij} = \frac{|c_j \cap p_i|}{|p_i|}.$$

Both the indexes range within the interval $[0, 1]$. The adequacy of a cluster c_j to an ideal partition p_i is measured by a combination of the above measures: the higher the values of the local quality indexes, the higher the correspondence between c_j and p_i.

- *global quality indexes* are useful to measure the overall quality of the clusters. In particular, a global quality index measures the overall fitting of a clustering scheme within an ideal partition. Assuming that each cluster c_j in the resulting clustering scheme is associated with an ideal partition $p_{h(j)}$ according to a criterion $h(j)$, *Eureka*! computes the *error rate* of the clustering scheme, i.e. the number of instances that are misclassified, weighted by the discrepancy between the expected number of partitions and the number of actual clusters:
  $$E = \gamma \times \frac{\sum_j \sum_{i \neq h(j)} m_{ij}}{\sum_{i,j} m_{ij}}$$
  where
  $$\gamma = \begin{cases} k/s & \text{if } k \geqslant s \\ s/k & \text{otherwise.} \end{cases}$$
  The index E is parametric to the association function h, which can be defined in different ways (for example, by exploiting local quality indexes). In its simplest form (implemented in *Eureka*!), the function $h(j)$ is computed by associating a cluster c_j with a partition $p_{h(j)}$ such that most of its instances are associated with that partition:
  $$h(j) = argmax_i \, m_{ij}.$$
  The main idea here is to measure the degree of agreement between the clustering partitioning and the ideal partitioning. Practically, we assume that a given cluster represents the partition that is mostly populated within the cluster itself, and consider the elements assigned to that cluster that do not belong to such a partition as errors. Thereby, better quality clustering results should produce almost diagonal contingency tables (up to permutation of the columns): the

higher the error rate, the more likely the discovered clustering scheme is to be rejected.

The computation of such measures can be easily obtained by appropriately selecting in the clustering pane the option to compare the cluster results with a predefined class attribute. Fig. 18(i) shows the visualization of the resulting contingency table. Once the clusters have been computed, the results of the algorithm are compared with the predefined classes, and the result of their comparison is shown in the "Clusterer output" pane, as shown in the above figure. In particular, the system displays the contingency table, the cluster assignments and the error rate.

3.3.2. Decision-tree based validation and interpretation

When no predefined structure is available (as it usually is the case), a possibility is to evaluate clustering results using only quantities and features inherent to the data set (see [35,36] for a comprehensive survey). For example, one can evaluate the global quality of a clustering scheme by measuring both its *compactness* (i.e. how close the elements of each cluster are to each other) and *separation* (i.e. how different are two distinct clusters). Again, various definitions of compactness and separation are possible. For example, we can measure the compactness by looking at the minimal (or average) intra-cluster similarity, and the separation by looking at the maximal extra-cluster similarity. As usual, such indexes can be used to evaluate and compare clustering results: "bad" clusters exhibit unsatisfying values of such indexes, while "good" clusters exhibit high values.

However, traditional approaches based on statistical indexes suffer of two main drawbacks. First, most of these methods hardly allow to compare different clustering schemes, obtained from different datasets that exhibit inherently different features: for example, if two datasets exhibit different data distributions, it is hard to compare objectively the results by means of compactness and separation. A sparse dataset is typically "less" compact than a dense dataset; nevertheless, the computed clustering schemes can be appropriate enough. Moreover, these methods generally fail in comparing different clustering algorithms. For example, it is not significant to compare algorithms that use different definitions of similarity (or distance).

Second and more importantly, often statistical indexes are not suitable to obtain an interpretation of the resulting clusters. A statistical index provides a criterion for accepting or rejecting a cluster, but does not provide enough information useful to characterize that cluster. As a consequence, one has to resort to further descriptive statistics (such as histogram distributions, modal values, etc.) in order to understand the meaning of the clustering scheme.

In *Eureka*!, in order to evaluate the validity of a clustering scheme when no further information is available, we adopted a different criterion which allows both to obtain both a quality index (useful to evaluate the validity of the clustering scheme) and a description of each cluster (useful to interpret each cluster). Such a criterion is mainly based on the observation that good clusters should be in clearly separable regions, and hence a classifier could easily characterize them. Practically, a clustering scheme in which the data points in a given cluster can be separated from the data points in the other clusters can be evaluated as a good clustering scheme (and, by the converse, a clustering scheme in which data points within clusters cannot be easily separated is a bad clustering scheme). Thus, the assessment of the validity of a clustering scheme can be done by means of a predictor of the cluster label in the original dataset. The prediction error hence becomes the criterion for assessing the validity of the scheme, under the hypothesis that a good clustering result should produce a low-error classifier. Notice that the approach has the additional advantage of being completely independent from the ‘‘structural’’ features of the datasets and of the algorithm being analyzed: since the evaluation criterion is mainly based on the capability of characterizing the results of a clustering scheme, the approach can be well-suited to compare both the results of different datasets and the results of different clustering schemes.

Many classification schemes can be applied to the purpose of building a predictor for the given clustering. In particular, decision-tree classifiers [37] can be well-suited for identifying linearly separable regions. More importantly, a decision-tree classifier provides a set of rules (directly obtained from the navigation of the classification tree), that directly provide an interpretation of each cluster resulting from the application of the clustering algorithm.

On the basis of the above considerations, the approach followed in *Eureka*! to validate and interpret a clustering scheme when no further information is available, is the following:

- For the given clustering scheme, a classification tree associated with such a scheme is built. The training set for this tree is the original dataset in which each tuple is labelled by the cluster of membership. Each of the original attributes is used as a predictor variable, and the cluster membership is used as a target variable.
- The prediction error of the resulting classification tree is evaluated. If the tree has a low error rate, the clustering is evaluated as a good clustering. If, on the contrary, the resulting classification tree has a high error rate, the clustering is evaluated as a bad clustering.
- If a low-error classification tree is obtained, then a set of rules is computed from the tree. Such rules provide the interpretation of the clustering scheme.

In order to implement such a validation scheme, we exploited the functionalities available in Weka. Weka cluster interface, in fact, allows to automatically add a cluster label (a new `cluster` attribute) to the dataset under consideration. Fig. 17(i), for example, shows the dataset resulting from the clustering phase. The capabilities provided by the process modelling of *Eureka*! further facilitate the cooperation between the clustering and the classification algorithm. Starting from this labelled data set in fact, we can set up a new knowledge discovery process in which we include a decision tree classifier with the aim of predicting the `cluster` attribute. The application of a decision-tree classification algorithm (e.g. the C4.5 algorithm) ends up with a tree representing the interpretation of the cluster partition, and a set of measures associated with the tree (such as percentage of correctly classified instances, mean error rate, etc.), that assess the validity of the clustering scheme obtained so far. Fig. 17(ii) shows an example result of the classification phase.

4. Experimental results

The described clustering methodology was extensively experimented in [38,39]. However, in both papers we discussed experiences done using the same approach but without using the *Eureka*! tool. Thus the methodology was applied step-by-step without the use of an integrated environment. After that we designed and implemented *Eureka*! as a high-level interactive tool for supporting users in all the steps of the KDD process. We used the *Eureka*! tool in several data mining experiments on both real and "academic" data from the UCI repository [40]. In the following we describe two such experiments, concerning respectively socio-economic and scientific data. In these experiments, we illustrate how the *Eureka*! methodology can be profitably adopted to interactively exploit domain information during the process of cluster generation, and assess the validity of the proposed approach by analyzing the results of cluster validation.

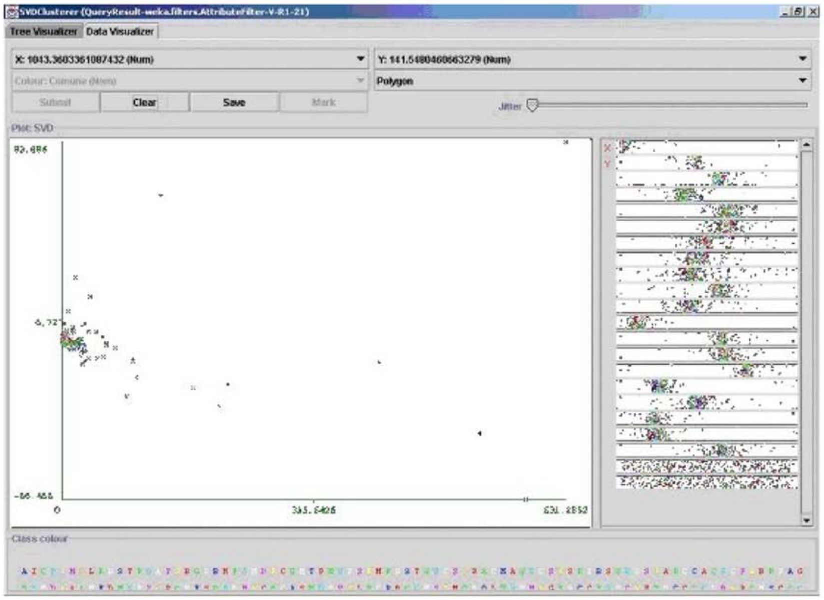
4.1. Socio-economic data

Recently we used *Eureka*! on a data set of *social-economic data* representing a collection of measurements made in 409 cities in an Italian district, and concerning social-economic factors such as unemployment rate, amount of companies, amount of agencies, etc. The main objective of the experiment was to detect clusters of cities which share a similar behavior as recipient of *e*-services. The resulted knowledge discovery process produced very interesting results about differentiating business proposals among the detected clusters [41]. In order to pursue such an objective, we first applied the interactive clustering methodology in order to find a suitable decomposition of the dataset, and next exploited the C4.5 decision tree algorithm in order to validate the decomposition and to obtain suitable descriptions of the given classes.

The experiment is particularly significant, since the user interaction in the clustering procedure can be guided by domain knowledge. Indeed, background knowledge can help in evaluating the consistency of the selection at each step of the clustering process, according to the interpretation we are able to provide about the detected clusters. Once we detect a relevant cluster, we can investigate the composition of such cluster and evaluate whether the cluster represents either an expected result, or an unexpected grouping that can be explained in terms of our prior knowledge about the cities that are involved.

Fig. 13(i) provides a two-dimensional representation of the (transformed) dataset. Each point in the image represents a city, and its position in the space depends from its attributes: closer points represent cities sharing similar features. In the figure we can detect

- a significant concentration of points (402 instances) towards the middle of the Y axis, and
- a dispersion of the remaining points (7 instances) in the rest of the plane. Interestingly, such instances represent the economically most relevant cities in the district.


SVDClusterer (QueryResult-weka.filters.AttributeFilter-V-R1-21)

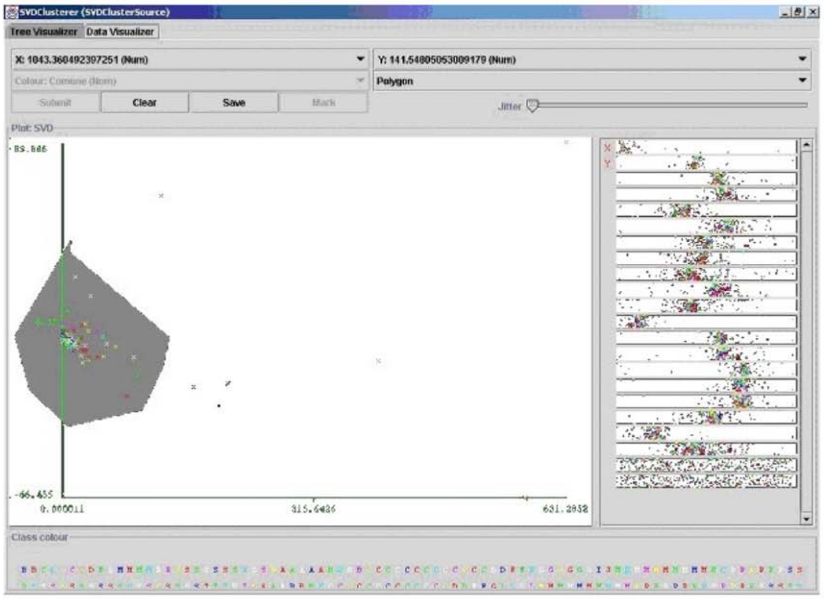
X: 1043.3603361087432 (Num)

Y: 141.5480460663279 (Num)

Colour: Comune (Nom)

Polygon

(i)


SVDClusterer (SVDClusterSource)

X: 1043.360492397251 (Num)

Y: 141.54805053009179 (Num)

Colour: Comune (Nom)

Polygon

(ii)

Fig. 13. Analysis of socio-economic data. (i) The visualization of the principal components exhibits a concentration towards the middle of the Y axis. (ii) Cluster separation: the concentration is separated from the other points in the plane.

The above observations suggest a separation of the points in two main groups, where the first group contains all the points belonging to the concentration, and the second group the remaining points. Fig. 13(ii) shows the separation of the two regions.

As already mentioned, the second group represents influential cities, and does not need to be further investigated. The clustering technique is then applied recursively in the cluster containing 402 instances. The visualization of such cluster does not put in evidence significant separations: a repeated application of the SVD decomposition to such a subset, however, highlights an interesting property holding in the subset. In Fig. 14(i), in fact, we can detect some unusual strips, parallel to the X axis, and exhibiting significant concentrations of points. The selection of such strips produces further clusters. Each strip can be analyzed separately, again producing further clusters: for example, the bottom-most strip can be further analyzed as shown in Fig. 14(ii).

Proceeding by subsequent partitioning steps, we obtained the final tree shown in Fig. 15(i), representing 10 clusters populated respectively by 17, 32, 64, 15, 93, 57, 25, 67, 32 and 7 instances. Interestingly, each population corresponded to an homogeneous group of cities sharing common features. For example, the group of 7 instances, detected in the first step, represented the most significant cities (from an economic point of view) in the district under consideration. Other groups represented cities sharing some specific peculiarities (such as the presence of harbor docks, or the co-location into depressed areas).

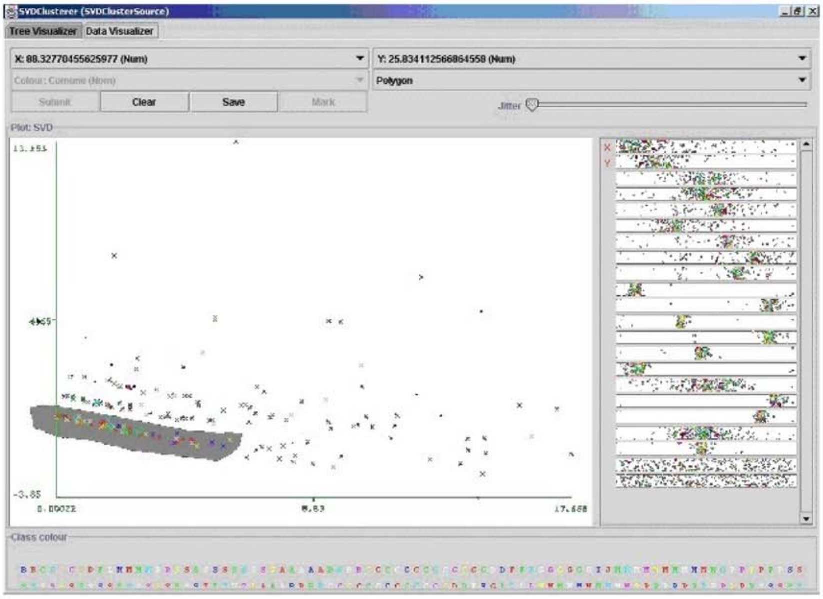
The results of the validation process is shown in Fig. 15(ii). As we can see, the error rate of the resulting classification tree is quite low: only 8 cases, out of a total of 409, are misclassified. Moreover, the rules extracted from the decision tree are quite easy to interpret, thus providing an immediate description of the contents of each cluster. For example, according to the decision tree, cluster 8 is characterized by the following pattern:

$$\begin{aligned}
&([\mathit{Business\ agencies}]\\
&\quad = 0 \wedge\ [\mathit{Working\ Professionals}] > 5 \wedge [\mathit{Insurance\ and\ Credit\ Services}] > 4)\\
&\vee\\
&([\mathit{Business\ Agencies}] > 1 \wedge [\mathit{Working\ Professionals}] > 14 \wedge\\
&\qquad [\mathit{Computer\ Services}] < = 6)
\end{aligned}$$

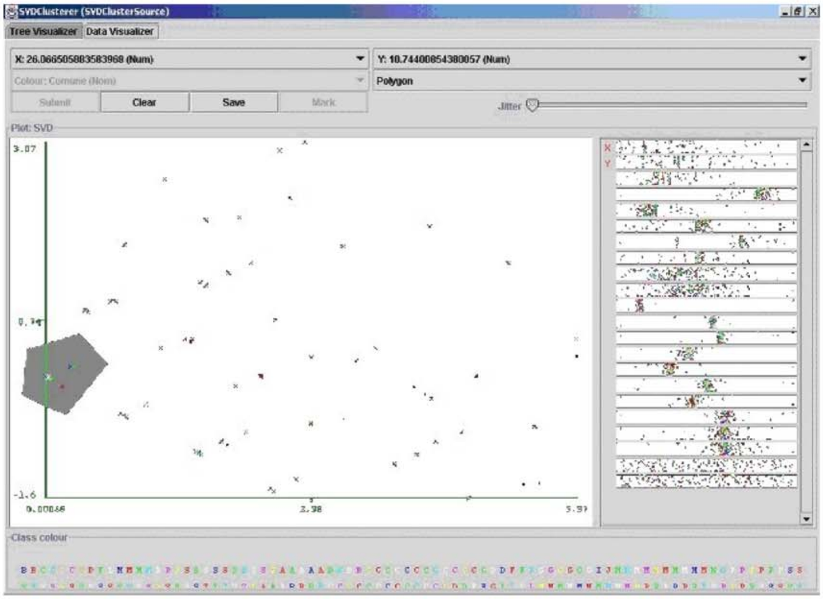
This rule identifies a class that includes small and medium towns with a limited set of professionals, enterprisers, and agencies potentially interested to *e*-services.

4.2. Scientific datasets

We now discuss the results obtained on the *segment* data set used in the previous sections as a running example. The experiment is useful mainly to illustrate the significance of the proposed methodology w.r.t. an external criteria approach. The dataset used in our experimentation contains 2310 instances, representing

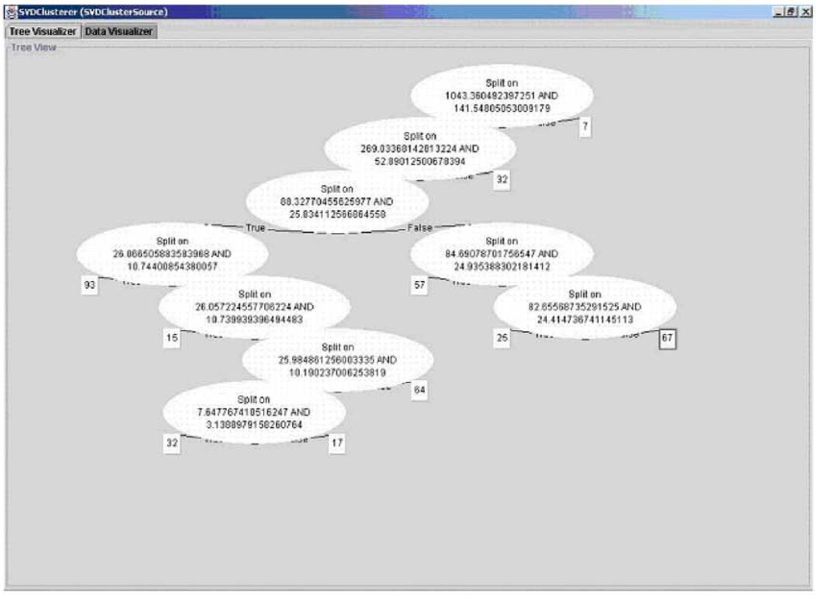


(i)

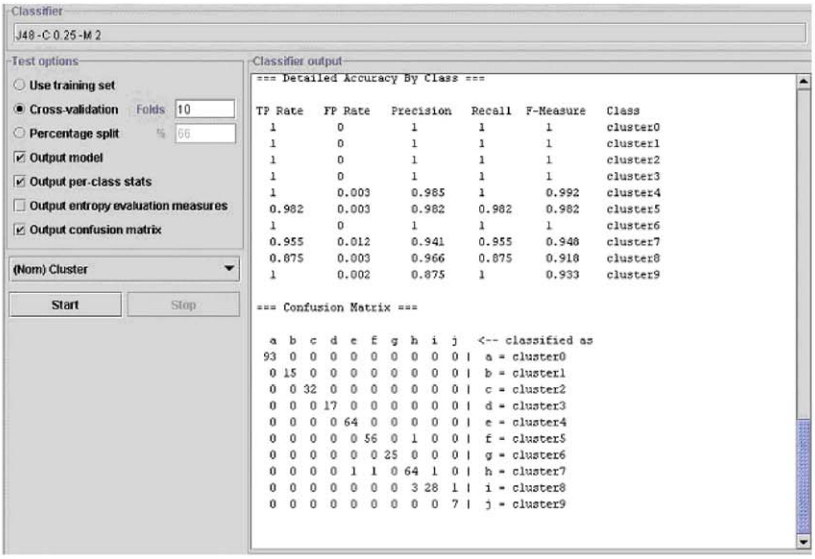


(ii)

Fig. 14. Analysis of socio-economic data. (i) The SVD decomposition of a subnode allows the visualization of interesting patterns: three strips are clearly visible, parallel to the X axis. One such strip is separated from the others and (ii) analyzed apart, exhibiting again a concentration of points and a dispersion of the remaining points.



(i)



Classifier: J48 -C 0.25 -M 2

Test options: Cross-validation [✓] Folds 10; Percentage split % 66

[✓] Output model
[✓] Output per-class stats
[ ] Output entropy evaluation measures
[✓] Output confusion matrix

(Nom) Cluster

```
=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  1         0          1          1         1        cluster0
  1         0          1          1         1        cluster1
  1         0          1          1         1        cluster2
  1         0          1          1         1        cluster3
  1         0.003      0.985      1         0.992    cluster4
  0.982     0.003      0.982      0.982     0.982    cluster5
  1         0          1          1         1        cluster6
  0.955     0.012      0.941      0.955     0.948    cluster7
  0.875     0.003      0.966      0.875     0.918    cluster8
  1         0.002      0.875      1         0.933    cluster9

=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j   <-- classified as
 93  0  0  0  0  0  0  0  0  0 |  a = cluster0
  0 15  0  0  0  0  0  0  0  0 |  b = cluster1
  0  0 32  0  0  0  0  0  0  0 |  c = cluster2
  0  0  0 17  0  0  0  0  0  0 |  d = cluster3
  0  0  0  0 64  0  0  0  0  0 |  e = cluster4
  0  0  0  0  0 56  0  1  0  0 |  f = cluster5
  0  0  0  0  0  0 25  0  0  0 |  g = cluster6
  0  0  0  0  1  1  0 64  1  0 |  h = cluster7
  0  0  0  0  0  0  0  3 28  1 |  i = cluster8
  0  0  0  0  0  0  0  0  0  7 |  j = cluster9
```

(ii)

Fig. 15. Analysis of socio-economic data. (i) Final cluster tree, and (ii) error rate of the validating decision tree.

respectively 7 classes: *brickface*, *cement*, *foliage*, *grass*, *path*, *sky* and *windows*. The classes are uniformly distributed in the dataset.

In the visual clustering process, we identified 17 clusters in the data set, as detailed in Fig. 16. The contingency table associated with such a partition is shown in Table 1. From the matrix we can see that almost each class has a corresponding cluster. In particular,

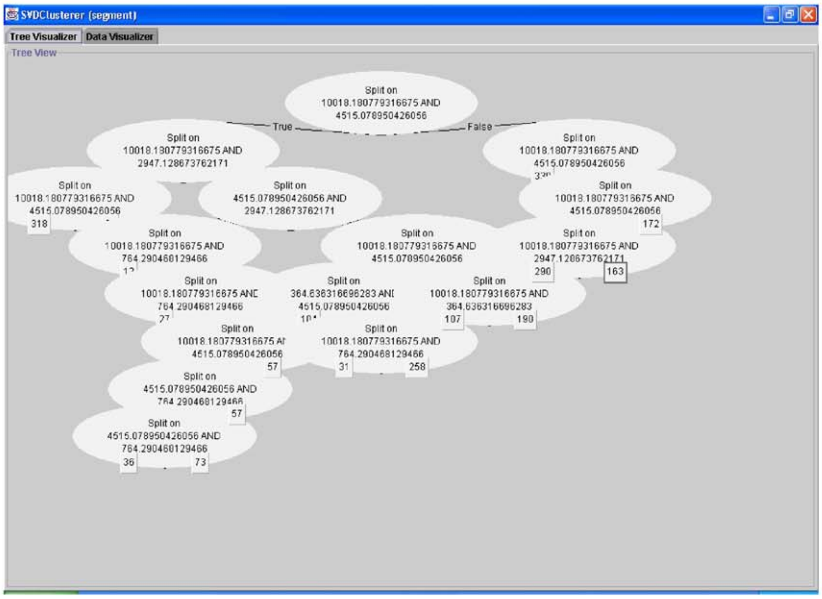
- clusters 8 and 12 represent *brickface*;
- clusters 15 and 16 represent *cement*;
- cluster 10 represents *foliage*;
- cluster 0 represents *grass*;
- cluster 14 represents *path*;
- cluster 13 represents *sky*;
- clusters 2,3,5,7, and 9 represent *windows*.

Most of the generated clusters are strongly biased towards a class (with the only exception of clusters 1, 6 and 11). As a consequence, the error rate corresponding to the clustering scheme is quite low (29%, where the actual misclassification error computed without considering the γ component is 12%). A decision-tree classifier trained to predict the `cluster` attribute produced a tree with a degree of accuracy of 94% and produced 77 rules describing the features of the discovered clusters. Fig. 17 details the analysis accomplished with such a decision tree.

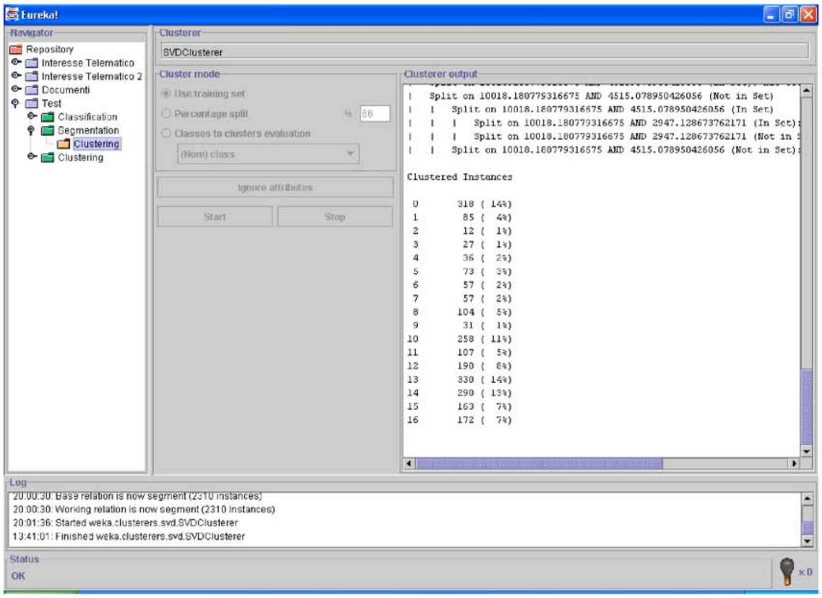
As a final remark, it is interesting to see how different clustering schemes can be compared according to such a validation methodology. To this purpose, we exploited the functionalities offered by the Weka environment, with the additional aim of comparing the results of the clustering algorithm implemented in *Eureka*! with the results of a different clustering algorithm. In particular, we compared such results with the clustering scheme resulting from the application of the EM algorithm [42] on the same data set. By imposing 7 classes (the optimal number of clusters, obtained via cross validation), we obtained an error rate of 45%, which is clearly a too high error rate (and thus it identifies a bad clustering, at least compared to the previous clustering scheme). The results the application of the algorithm are shown in Fig. 18(ii). It is worth noticing how different parameter tuning in EM (e.g. different class numbers) produced even a less accurate classification.

Table 1
Contingency table for *segment*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Brickface					31	11			102		5	1	180				
Cement	5	22					1	1	1	1	5	9	5			124	156
Foliage						5	38	5		4	199	53	4			6	16
Grass	309	21															
Path		40													290		
Sky														330			
Window	4	2	12	27	5	57	18	51	1	26	49	44	1			33	


SVDClusterer (segment)
Tree Visualizer
Data Visualizer
Tree View
Split on 10018.180779316675 AND 4515.078950426056
True
False
Split on 10018.180779316675 AND 2947.128673762171
Split on 10018.180779316675 AND 4515.078950426056
Split on 10018.180779316675 AND 4515.078950426056
318
Split on 4515.078950426056 AND 2947.128673762171
Split on 10018.180779316675 AND 4515.078950426056
172
Split on 10018.180779316675 AND 764.290468129466
Split on 10018.180779316675 AND 4515.078950426056
Split on 10018.180779316675 AND 2947.128673762171
290
163
Split on 10018.180779316675 AND 764.290468129466
Split on 364.636316696283 AND 4515.078950426056
Split on 10018.180779316675 AND 364.636316696283
107
190
Split on 10018.180779316675 AND 4515.078950426056
57
Split on 10018.180779316675 AND 764.290468129466
31
258
Split on 4515.078950426056 AND 764.290468129466
57
Split on 4515.078950426056 AND 764.290468129466
36
73

(i)

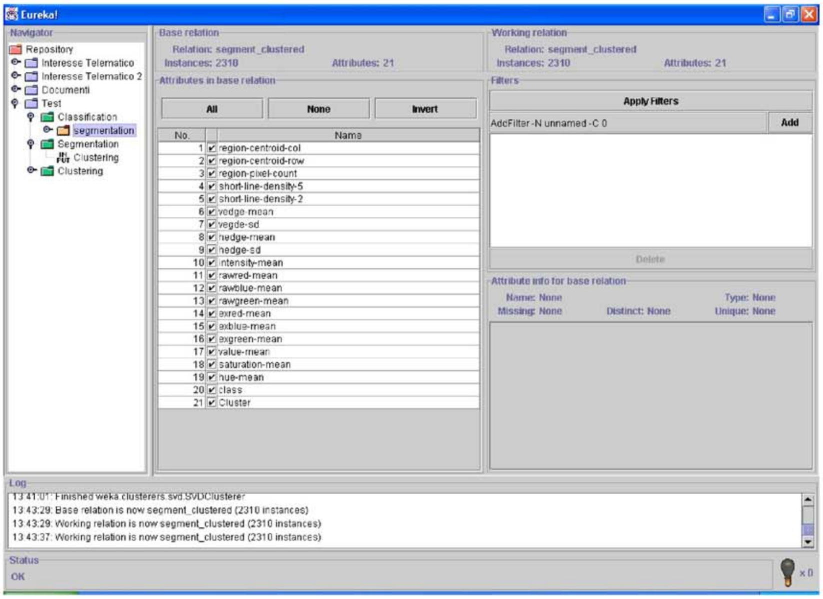


```
|   Split on 10018.180779316675 AND 4515.078950426056 (Not in Set)
|   |   Split on 10018.180779316675 AND 4515.078950426056 (In Set)
|   |   |   Split on 10018.180779316675 AND 2947.128673762171 (In Set):
|   |   |   Split on 10018.180779316675 AND 2947.128673762171 (Not in S
|   |   Split on 10018.180779316675 AND 4515.078950426056 (Not in Set):

Clustered Instances

 0        318 ( 14%)
 1         85 (  4%)
 2         12 (  1%)
 3         27 (  1%)
 4         36 (  2%)
 5         73 (  3%)
 6         57 (  2%)
 7         57 (  2%)
 8        104 (  5%)
 9         31 (  1%)
10        258 ( 11%)
11        107 (  5%)
12        190 (  8%)
13        330 ( 14%)
14        290 ( 13%)
15        163 (  7%)
16        172 (  7%)
```

```
20:00:30: Base relation is now segment (2310 instances)
20:00:30: Working relation is now segment (2310 instances)
20:01:36: Started weka.clusterers.svd.SVDClusterer
13:41:01: Finished weka.clusterers.svd.SVDClusterer
```

(ii)

Fig. 16. Analysis of the *segment* database. (i) Final Cluster Tree. The leaves of the tree represent the final partition of the origin dataset (represented by the root). (ii) Final distribution of the dataset in the leaf nodes.



Base relation
Relation: segment_clustered
Instances: 2310 Attributes: 21

Working relation
Relation: segment_clustered
Instances: 2310 Attributes: 21

Attributes in base relation

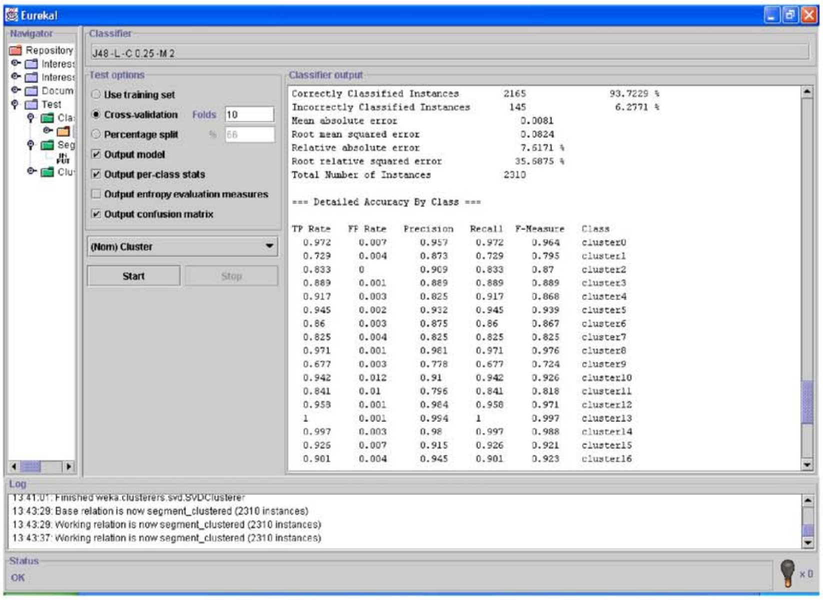
No.		Name
1	✓	region-centroid-col
2	✓	region-centroid-row
3	✓	region-pixel-count
4	✓	short-line-density-5
5	✓	short-line-density-2
6	✓	vedge-mean
7	✓	vegde-sd
8	✓	hedge-mean
9	✓	hedge-sd
10	✓	intensity-mean
11	✓	rawred-mean
12	✓	rawblue-mean
13	✓	rawgreen-mean
14	✓	exred-mean
15	✓	exblue-mean
16	✓	exgreen-mean
17	✓	value-mean
18	✓	saturation-mean
19	✓	hue-mean
20	✓	class
21	✓	Cluster

Filters
AddFilter -N unnamed -C 0

Attribute info for base relation
Name: None Type: None
Missing: None Distinct: None Unique: None

Log
```
13:41:01: Finished weka.clusterers.svd.SVDClusterer
13:43:29: Base relation is now segment_clustered (2310 instances)
13:43:29: Working relation is now segment_clustered (2310 instances)
13:43:37: Working relation is now segment_clustered (2310 instances)
```
Status
OK

(i)



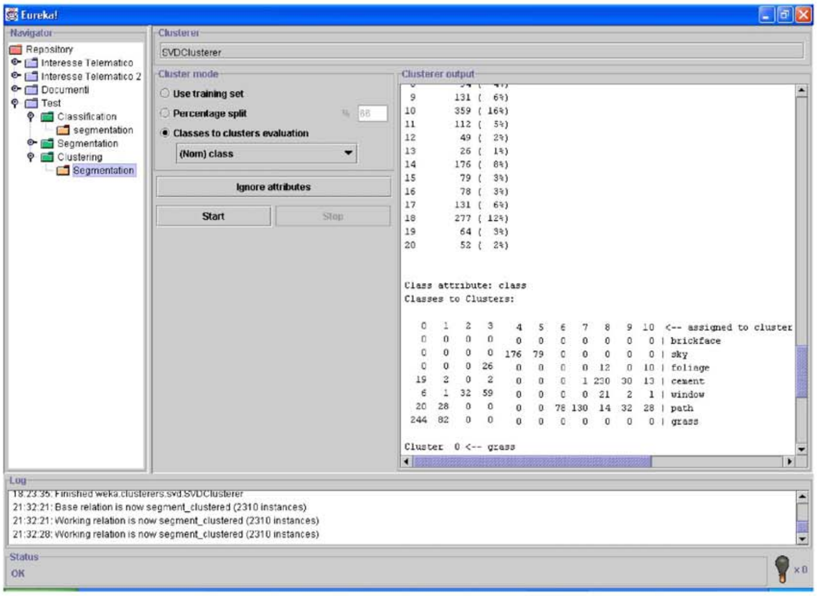
Classifier
J48 -L -C 0.25 -M 2

Test options
Use training set
[✓ Cross-validation] Folds 10
Percentage split % 66
[✓] Output model
[✓] Output per-class stats
Output entropy evaluation measures
[✓] Output confusion matrix
(Nom) Cluster

Classifier output
```
Correctly Classified Instances        2165               93.7229 %
Incorrectly Classified Instances       145                6.2771 %
Mean absolute error                      0.0081
Root mean squared error                  0.0824
Relative absolute error                  7.6171 %
Root relative squared error             35.6875 %
Total Number of Instances             2310

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.972     0.007      0.957     0.972     0.964    cluster0
  0.729     0.004      0.873     0.729     0.795    cluster1
  0.833     0          0.909     0.833     0.87     cluster2
  0.889     0.001      0.889     0.889     0.889    cluster3
  0.917     0.003      0.825     0.917     0.868    cluster4
  0.945     0.002      0.932     0.945     0.939    cluster5
  0.86      0.003      0.875     0.86      0.867    cluster6
  0.825     0.004      0.825     0.825     0.825    cluster7
  0.971     0.001      0.981     0.971     0.976    cluster8
  0.677     0.003      0.778     0.677     0.724    cluster9
  0.942     0.012      0.91      0.942     0.926    cluster10
  0.841     0.01       0.796     0.841     0.818    cluster11
  0.958     0.001      0.984     0.958     0.971    cluster12
  1         0.001      0.994     1         0.997    cluster13
  0.997     0.003      0.98      0.997     0.988    cluster14
  0.926     0.007      0.915     0.926     0.921    cluster15
  0.901     0.004      0.945     0.901     0.923    cluster16
```
Log
```
13:41:01: Finished weka.clusterers.svd.SVDClusterer
13:43:29: Base relation is now segment_clustered (2310 instances)
13:43:29: Working relation is now segment_clustered (2310 instances)
13:43:37: Working relation is now segment_clustered (2310 instances)
```
Status
OK

(ii)

Fig. 17. Analysis of the *segment* database. (i) Each originary instance in the dataset is labelled by the cluster it is assigned to. (ii) Cluster validation: a decision tree is trained over the labelled instances, and the error rate is evaluated.



Clusterer: SVDClusterer

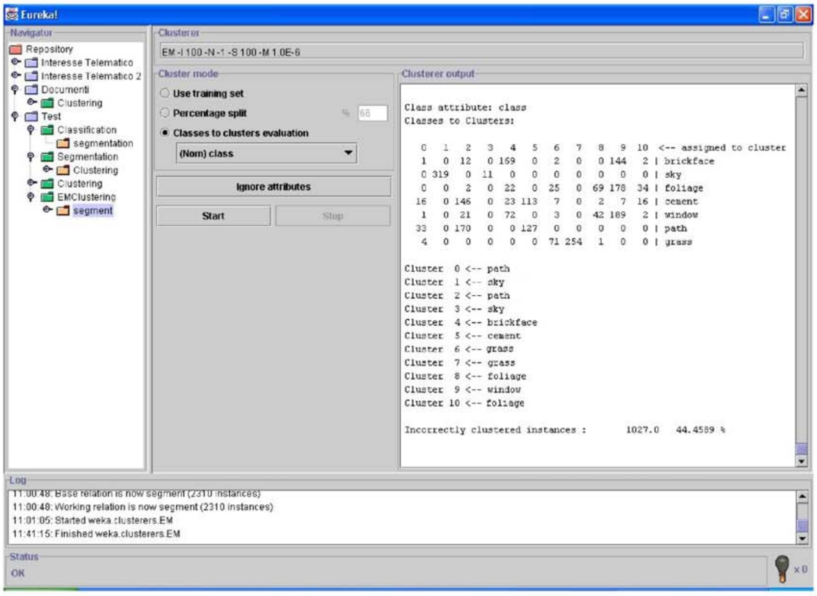
Cluster mode: Classes to clusters evaluation — (Nom) class

```
 9         131 (  6%)
10         359 ( 16%)
11         112 (  5%)
12          49 (  2%)
13          26 (  1%)
14         176 (  8%)
15          79 (  3%)
16          78 (  3%)
17         131 (  6%)
18         277 ( 12%)
19          64 (  3%)
20          52 (  2%)


Class attribute: class
Classes to Clusters:

   0   1   2   3    4   5   6   7   8   9  10  <-- assigned to cluster
   0   0   0   0    0   0   0   0   0   0   0 | brickface
   0   0   0   0  176  79   0   0   0   0   0 | sky
   0   0   0  26    0   0   0   0  12   0  10 | foliage
  19   2   0   2    0   0   0   1 230  30  13 | cement
   6   1  32  59    0   0   0   0  21   2   1 | window
  20  28   0   0    0   0  78 130  14  32  28 | path
 244  82   0   0    0   0   0   0   0   0   0 | grass

Cluster  0 <-- grass
```

```
18:23:35: Finished weka.clusterers.svd.SVDClusterer
21:32:21: Base relation is now segment_clustered (2310 instances)
21:32:21: Working relation is now segment_clustered (2310 instances)
21:32:28: Working relation is now segment_clustered (2310 instances)
```

Status: OK

(i)



Clusterer: EM -I 100 -N -1 -S 100 -M 1.0E-6

Cluster mode: Classes to clusters evaluation — (Nom) class

```
Class attribute: class
Classes to Clusters:

  0   1   2   3   4   5   6   7   8   9  10  <-- assigned to cluster
  1   0  12   0 169   0   2   0   0 144   2 | brickface
  0 319   0  11   0   0   0   0   0   0   0 | sky
  0   0   2   0  22   0  25   0  69 178  34 | foliage
 16   0 146   0  23 113   7   0   2   7  16 | cement
  1   0  21   0  72   0   3   0  42 189   2 | window
 33   0 170   0   0 127   0   0   0   0   0 | path
  4   0   0   0   0   0  71 254   1   0   0 | grass

Cluster  0 <-- path
Cluster  1 <-- sky
Cluster  2 <-- path
Cluster  3 <-- sky
Cluster  4 <-- brickface
Cluster  5 <-- cement
Cluster  6 <-- grass
Cluster  7 <-- grass
Cluster  8 <-- foliage
Cluster  9 <-- window
Cluster 10 <-- foliage

Incorrectly clustered instances :      1027.0   44.4589 %
```

```
11:00:48: Base relation is now segment (2310 instances)
11:00:48: Working relation is now segment (2310 instances)
11:01:05: Started weka.clusterers.EM
11:41:15: Finished weka.clusterers.EM
```

Status: OK

(ii)

Fig. 18. Analysis of the *segment* database. (i) Contingency table for the *segment* database. (ii) Contingency table using the EM algorithm.

5. Conclusions and future work

In this paper we described the main features of an interactive knowledge discovery tool, named *Eureka*!, which combines a visual clustering method to hypothesize meaningful structures in the data and a classification machine learning algorithm to validate the hypothesized structures. A two-dimensional representation of the available data allows users to partition the search space by choosing shape or density according to criteria he estimates optimal. The accuracy of clustering results obtained through the user intervention can be validated by using a decision tree classifier which is a component of the mining tool. We used a simple data set to describe the tool features and how the discovery process is performed using it.

The combination of clustering and classification techniques with an interactive visual approach is the main contribution of the paper. In fact, *Eureka*! is based on tightly coupling visualization and analytical mining into one data mining tool. Human decisions based on visualization allow analytical procedures to enlarge their scope and, at the same time to help humans in dealing with decisions that can no longer be automated. This results in a tightly coupled visual data mining environment that guides users through the mining process from data source to knowledge models. The test cases analyzed in Section 4 showed that a visual clustering methodology can produce more accurate results compared to that obtained using an oblivious clustering algorithm.

Currently we are using *Eureka*! to mine different data sets in several application domains. At the same time, we are working on tool improvements and extensions such as in/out zooming features and automatic region separation suggestions provided to users by the system at each splitting step.

References

[1] U.M. Fayyad, G. Piatesky-Shapiro, P. Smith, From data mining to knowledge discovery: an overview, in: U. Fayyad, et al. (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, MA, 1996, pp. 1–34.
[2] D.A. Keim, S. Eick, Proceedings of the ACM SIGKDD, Workshop on Visual Data Mining, 2001.
[3] D.A. Keim, Information visualization and visual data mining, IEEE Transactions on Visualization and Computer Graphics 8 (1) (2002) 1–8.
[4] U. Fayyad, G.G. Grinstein, A. Wierse, Information Visualization in Data Mining and Knowledge Discovery, Morgan Kaufmann, Los Altos, CA, 2001.
[5] L. Yang, Interactive exploration of very large relational datasets through 3d dynamic projections, in: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 236–243.
[6] K. Fishkin, M.C. Stone, Enhanced dynamic queries via movable filters, in: Proceedings of the ACM CHI '95 Conference, 1995, pp. 415–420.
[7] R. Rao, S.K. Card, The table lens: merging graphical and symbolic representation in an interactive focus + context visualization for tabular information, in: Proceedings of the ACM CHI '94 Conference, 1994, pp. 318–322.
[8] M.O. Ward, Xmdvtool: integrating multiple methods for visualizing multivariate data, in: Proceedings of the IEEE Visualization Conference, 1994, pp. 326–336.

[9] A. Hinnenburg, D.A. Keim, M. Wawryniuk, Hd-eye: visual mining of high dimensional data, IEEE Computer Graphics and Applications 19 (5) (1999) 22–31.
[10] C.C. Aggarwal, A human computer cooperative system for effective high dimensional clustering, in: Proceedings of 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 221–226.
[11] E. Kandogan, Visualizing multi-dimensional clustering, trends, and outliers using star coordinates, in: Proceedings of 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 107–116.
[12] F. Poulet, Cooperation between automatic algorithms, interactive algorithms and visualization tools for visual data mining, in: Proceedings of the PKDD '02 International Workshop on Visual Data Mining, 2002, pp. 67–79.
[13] M. Ankerst, M. Ester, A. Kriegel, Visual classification: an interactive approach to decision tree construction, in: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 392–396.
[14] M. Ankerst, M. Ester, H.P. Kriegel, Towards an effective cooperation of the computer and the user for classification, in: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 179–188.
[15] H. Hoffman, A. Siebes, A. Wilhelm, Visualizing association rules with interactive mosaic plots, in: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 227–235.
[16] A. Hinnenburg, D.A. Keim, Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering, in: Proceedings of the 25th International Conference on Very Large DataBases, 1999, pp. 506–517.
[17] M. Ankerst, M. Breunig, H.P. Kriegel, J. Sandler, OPTICS: ordering points to identify the clustering structure, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1999, pp. 49–60.
[18] K.R. Muller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf, An introduction to kernel-based learning algorithms, IEEE Transactions on Neural Networks 12 (2) (2001) 181–201.
[19] G. Strang, Linear Algebra and its Applications, Academic Press, New York, 1980.
[20] G. Manco, C. Pizzuti, D. Talia, Eureka!: a tool for interactive knowledge discovery, in: Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA 2002), Lecture Notes in Computer Science 2453 (2002), pp. 381–391.
[21] I. Witten, E. Frank, Data Mining: Practical Machine Learning Tools with Java Implementation, Morgan Kaufman, Los Altos, CA, 1999.
[22] E. Beltrami, Sulle funzioni bilineari [on bilinear functions], Giornale di Matematiche ad Uso degli Studenti delle Università 11 (1873) 98–106.
[23] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[24] I.T. Jolliffe, Principal Component Analysis, Springer, Berlin, 1986.
[25] F. Korn, H.V. Jagadish, C. Faloutsos, Efficient supporting ad hoc queries in large datasets of time sequences, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1997, pp. 289–300.
[26] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Receipes in C: The Art of Computing, Cambridge University Press, Cambridge, 1992.
[27] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley–ACM Press, New York, 1999.
[28] G.H. Golub, C.F. Van Loan, Matrix Computation, The Johns Hopkins University Press, Baltimore, MD, 1989.
[29] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[30] B. Everitt, Cluster Analysis, Heinemann Educational Books Ltd., London, 1977.
[31] L. Kaufman, P.J. Rousseew, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, New York, 1990.
[32] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufman, Los Altos, CA, 2000.

[33] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On clustering validation techniques, Journal of Intelligent Information Systems 17 (2–3) (2001) 107–145.

[34] S. Theodoridis, K. Koutroubas, Pattern Recognition, Academic Press, New York, 1999.

[35] M. Halkidi, Y. Batistakis, M. Vazirgiannis, Cluster validity methods: part I, SIGMOD Record 31 (2) (2002) 40–45.

[36] M. Halkidi, Y. Batistakis, M. Vazirgiannis, Cluster validity checking methods: part II, SIGMOD Record 31 (3) (2002) 19–27.

[37] R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, Los Altos, CA, 1993.

[38] E. Maltseva, C. Pizzuti, D. Talia, Visual data mining for defining tax-payer profiles, in: Proceedings of the International Workshop in Information retrieval (IR '2001), 2001, pp. 107–114.

[39] E. Maltseva, C. Pizzuti, D. Talia, Mining high-dimensional scientific data sets using singular value decomposition, in: R.L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, R. Namburu (Eds.), Data Mining for Scientific and Engineering Applications, Kluwer Academic Publishers, Dordrecht, 2001, pp. 425–438.

[40] C.L. Blake, C.J. Merz, UCI repository of machine learning, in: http://kdd.ics.uci.edu, University of California at Irvine, Department of Information and Computer Science.

[41] R. Bosco, G. Manco, C. Pizzuti, D. Talia, Data Mining per il Monitoraggio del Territorio: Il Caso di Studio del Piano Telematico Calabria, in: 39° Congresso annuale dell'Associazione Italiana per il Calcolo Automatico (AICA2001), 2001.

[42] G.J. MacLahan, T. Krishnan, The EM Algorithm and Extensions, Wiley, New York, 1997.