



POLITECNICO DI TORINO

M.Sc. in Computer Science Engineering

DEPARTMENT OF DAUIN

**Automated Data Exploration To Discover Structures
And Models Hidden In The Data**

Master's candidate:

Paolo BETHAZ

Supervisor:

Prof. Tania CERQUITELLI

october 2019

To all my family

Acknowledgements

Turin, october 2019

I would like to thank all the people who supported me writing this thesis. First of all I thank all my family, in particular my parents and my brother, who have always been close to me and helped me patiently to reach the end of this university career. A special thanks also to my supervisor professor Tania Cerquitelli, who gave me the opportunity to work on this very interesting topic, always following me with a lot of patience and availability.

Moreover, thanks to all my friends, the old ones and those most recently met along my journey, for the fun and carefree moments spent together in these years. I know I can always count on your good mood.

Last but not least, thanks to Stefania. Not only for having taught me to use Latex, but for always supporting me in this last period, sharing many joyful moments that allowed me to write this thesis with more serenity.

Contents

1	Introduction	7
1.1	Techniques Used	8
2	State of the art	9
2.1	Datawrapper	10
2.2	AnswerMiner	11
2.3	JetPack Data	13
2.4	DataPine	13
2.5	Dive	14
2.6	NCSS	15
2.7	ML Sandbox	17
3	Automated Data Exploration	19
3.1	Data type identification	20
3.2	Data characterization	20
3.3	Outlier detection	22
3.4	Self-tuning cluster analysis	24
3.4.1	K-Means Clustering	26
3.4.2	DBSCAN	28
3.4.3	Hierarchical Clustering	31
3.5	Classification	35
3.6	Data Transformation	37
3.7	Visualization	37

4	Architecture and Workflow	39
4.1	Data	41
4.2	Data Characterization	42
4.3	Exploratory and Unsupervised learning	43
4.4	Data Transformation	48
4.5	StoryTelling	51
5	Results and evaluation	56
5.1	Iris Dataset	56
5.1.1	Data	57
5.1.2	User Exploration	57
5.1.3	StoryTelling	61
5.2	Prescription Dataset	63
5.2.1	Data	64
5.2.2	User Exploration	64
5.2.3	Data Transformation	66
5.3	Storytelling	67
6	Conclusions	69

List of Figures

2.1	<i>First two steps of Datawrapper application</i>	11
2.2	<i>Third and fourth steps of Datawrapper application</i>	11
2.3	<i>General overview of the functionality offered by AnswerMiner</i>	12
2.4	<i>Layout of the Datapine analysis page</i>	14
2.5	<i>Four stages of a data exploration workflow in DIVE: (A) Datasets, (B) Visualize, (C) Analysis, and (D) Stories</i>	15
2.6	<i>Procedures offered by NCSS analysis</i>	16
3.1	<i>Traditional Knowledge Extraction Process</i>	19
3.2	<i>Automated Workflow proposed by the framework</i>	20
3.3	<i>Different parts of a boxplot [6]</i>	24
3.4	<i>Example of how the One Hot Encoding Works [9]</i>	25
3.5	<i>Result of the elbow method for choosing the best number of clusters [10]</i>	27
3.6	<i>Find the elbow of a curve, looking for the point with the greatest distance from the line b (between the first and the last point of the curve) [12]</i>	28
3.7	<i>Example of based density on the left. Core, border and noise points on the right [7]</i>	29
3.8	<i>Hierarchical Clustering using 'min' as a criterion of similarity between two clusters [7]</i>	32
3.9	<i>Hierarchical Clustering using 'max' as a criterion of similarity between two clusters [7]</i>	33
3.10	<i>Hierarchical Clustering using 'average' as a criterion of similarity between two clusters [7]</i>	33

3.11	<i>Hierarchical Clustering using 'ward' as a criterion of similarity between two clusters [7]</i>	34
4.1	<i>Map containing the different data exploration tasks offered by the framework</i>	39
4.2	<i>WorkFlow of the automated data exploration proposed by the framework</i>	40
4.3	<i>Upload Page</i>	41
4.4	<i>Radar Plot Schema [16]</i>	45
4.5	<i>Example of a Decision Tree applied to the Iris dataset [14][17]</i>	47
4.6	<i>The two attributes 'Id' and 'Exam' are merged into a single column</i>	50
4.7	<i>Example of how k-fold cross-validation works [18]</i>	53
4.8	<i>Example of confusion matrix with two labels (positive and negative). True positive (TP) and True negative (TN) are the values classified correctly [20]</i>	54
5.1	<i>Section 'Data'. First page shown. The user can scroll the table horizontally and vertically to see the whole dataset</i>	57
5.2	<i>Section 'Info'. The user can see some statistics on the various attributes</i>	58
5.3	<i>Section 'Info'. 'Species' is the only textual attribute. Histogram containing the number of elements for each species</i>	58
5.4	<i>Section 'Info'. Correlation Matrix</i>	59
5.5	<i>Section 'Info'. Boxplots for each numerical field</i>	59
5.6	<i>Section 'Info'. CDF and Histograms for each numerical field</i>	59
5.7	<i>Section 'Clustering'. 3d scatter chart to show the outliers</i>	60
5.8	<i>Section 'Clustering'. Elbow graph and 3d scatter chart for the K-means. The radar chart is obtained by clicking on the centroid2 on the scatter chart. To the right of the scatter chart there is also a pie chart</i>	60
5.9	<i>Section 'Clustering'. 3d scatter chart and pie chart for the DBSCAN. On the left of the scatter chart there is also a k-distance graph. The stacked bar chart is obtained clicking on the button 'Compare with original label'</i>	61
5.10	<i>Section 'Clustering'. Dendrogram and 3d scatter chart for the Hierarchical Clustering. The popup window is obtained by clicking on the button 'See decision tree'</i>	61
5.11	<i>Section 'Storytelling'. First Page</i>	62

5.12	<i>Section 'Storytelling'. First Page. Result obtained by clicking on the correlation matrix cell highlighted in the previous image</i>	62
5.13	<i>Section 'Storytelling'. Second Page</i>	62
5.14	<i>Section 'Storytelling'. Second Page. Result obtained by clicking on any point in the cluster0</i>	63
5.15	<i>Section 'Storytelling'. Third Page</i>	63
5.16	<i>Section 'Data'. First page after the upload page</i>	64
5.17	<i>Vertical navigation bar present on all pages. In this case there is also the section 'Data Transformation'</i>	65
5.18	<i>Section 'Clustering'. The structure of this section is the same as that reported for the Iris dataset. But in this case, since there is no label, the 'Compare with original label' button is no longer present</i>	66
5.19	<i>Section 'Data Transformation'</i>	66
5.20	<i>Section 'Data Transformation'. Result obtained after clicking on the transformation highlighted in the previous image</i>	67
5.21	<i>Section 'Storytelling'. First Page</i>	67
5.22	<i>Section 'Storytelling'. Second Page</i>	68
5.23	<i>Section 'Storytelling'. Third Page</i>	68
5.24	<i>Section 'Storytelling'. Third Page. Window opened after clicking on the transformation highlighted in the previous image</i>	68

List of Tables

2.1	<i>Dataset Schema</i>	9
4.1	<i>Initial Dataset</i>	49
4.2	<i>Dataset After Transformation</i>	49
5.1	<i>Iris Dataset</i>	57
5.2	<i>Prescription Dataset</i>	64

Chapter 1

Introduction

In today's world, data value is severely undermined by our inability to translate them into actionable knowledge. Up to now, a lot of research efforts have been devoted to enhancing the effectiveness and efficiency of analytics algorithms. However, their exploitation is still a multi-step process requiring a lot of expertise to be correctly performed. To streamline the knowledge extraction process and enhance the friendliness of data analytics task, what we tried to do is to design and develop a new generation of data analytics solutions to automatically discover descriptive, predictive and prescriptive models and structures hidden in the data without requiring the intervention of the data scientist. A key feature of such solutions is the automation of the full data-analytic workflow, from heterogenous data ingestion to the result presentation.

To do this, in this thesis work we implemented a new framework that aims to be as user-friendly as possible, so that it can be accessible by everyone. Furthermore, since our framework aims to analyze a dataset automatically, minimizing the contribution of the data scientist, the skills required by the user who uses it are minimal. We built it so that all users, even the less experienced in the field, can use it easily, going to meet the concept of democratizing data science.

1.1 Techniques Used

In implementing our new framework we have chosen to use python as programming language, which is one of the most used programming languages for Data Mining and Machine Learning techniques. In particular, the libraries and modules we used most to produce this work were the following:

- Pandas, which is a library for manipulating data in a tabular or sequential format. Very useful for loading and saving standard formats for tabular data, such csv and Excel files
- Matplotlib, a very powerful and flexible library for the creation of graphics, which allows the creation of many types of plots in a simple and intuitive way
- Seaborn, which is a matplotlib-based data visualization library. Seaborn provides a high-level interface for drawing information-rich statistical graphs
- Sklearn, which is an automatic learning library that contains all the most important classification, regression and clustering algorithms

To build the architecture of our web application we used Flask, which is a micro-framework for Python web applications[1]. The graphic part of our web pages is then managed thanks to Css and Javascript. Of particular interest was HighCharts [2], a library written in pure Javascript which allowed us to create lots of interactive and high-level graphics.

Chapter 2

State of the art

Before analyzing our application in detail, let's see and describe other existing applications that offer similar functionality to ours. In particular we have reported below only the free applications on the web, that any user can exploit without any restrictions. These applications obviously don't do exactly what we have tried to implement, but they still have common features useful for comparing our work. In order to better compare the various results obtained with these different tools, we tested each of these with the same dataset. The dataset used is 'prescrizioni.csv', a very simple clinical dataset that contains 500 medical prescriptions, each of these characterized by the id of the patient, the age of the patient, the sex of the patient, the date on which the prescription was made, a code that characterized the type of exam and a code that characterized the type of branca (which includes several exams). The schema of the dataset used is shown in Table 2.1.

id	eta	sess	codpresta	codbranca	data
31	72	1	89.7	99	20070214
31	72	1	90.27.1	98	20070214
31	72	1	90.44.4	98	20070214

Table 2.1: *Dataset Schema*

2.1 Datawrapper

Datawrapper is an innovative tool launched in 2012 which aims to allow users to create and visualize interactive charts and maps. Charts and maps are very useful and intuitive in data analysis, but creating them is often a very time-consuming operation, because requires someone with design experience and someone who knows the basic rules of charts. These long times could be a problem for example for a journalist or a researcher working under deadline. Datawrapper is proposed to be a very good application to use when speed is an essential requirement, creating charts and maps in few minutes.

This application can be shortly described in four different steps (see Figures 2.1 and 2.2):

- In the first step the user can copy and paste a data table or he can upload a csv/xls file
- In the second step, called 'check and describe', the user can see his entire dataset like a big table and he can sort the view according to the various attributes. The columns are shown in red if they are numerical attributes and are shown in black if they are text. Here the user can also decide to drop one or more columns from the dataset, or add new ones.
- In the third step the user can see his dataset in many different ways, he can choose from many chart and map types and after choosing one of these, he can customize it to make it more effective
- Finally, the user can export his chart as a png, svg or pdf document to print it

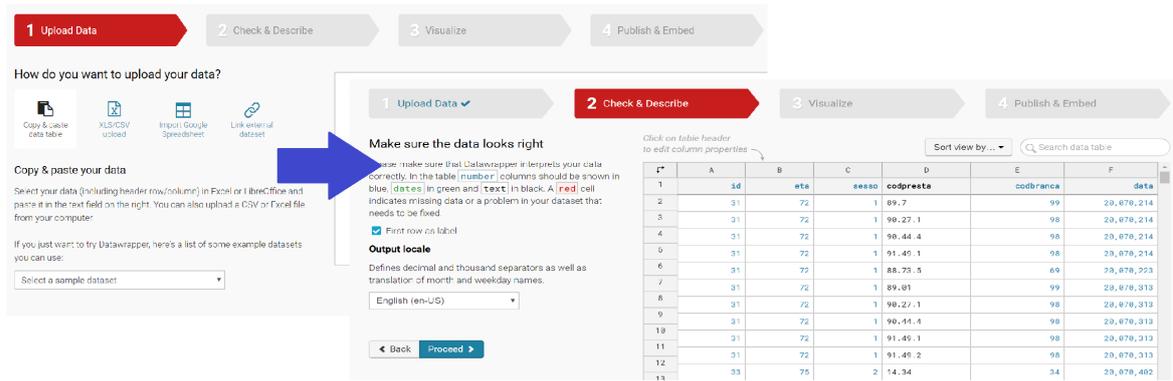


Figure 2.1: First two steps of Datawrapper application

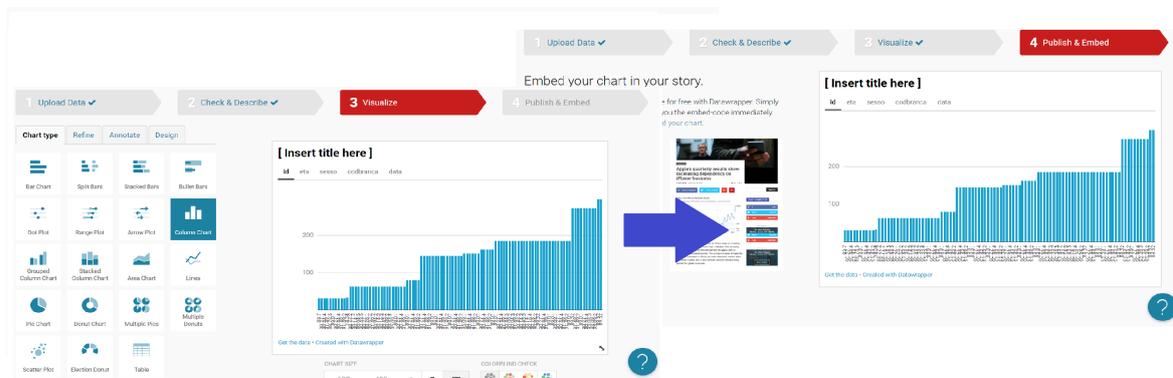


Figure 2.2: Third and fourth steps of Datawrapper application

2.2 AnswerMiner

AnswerMiner is a data exploration and visualization tool focused on the usability and simplicity. This software is designed to help everyone is working with data, from the expert data-scientist to the simple user that has no experience with data analysis. The key to ensuring this functionality is the very user-friendly interface that this tool offers, making the use of the features understandable to anyone. This application has a monthly cost to pay to be used, but there is also a free version with the limitation that the user can't analyze a dataset with more than 1000 rows and 20 columns. To test this tool and see in detail what features were offered, we used this free version. Now let's see what are the steps to follow to analyze a dataset with

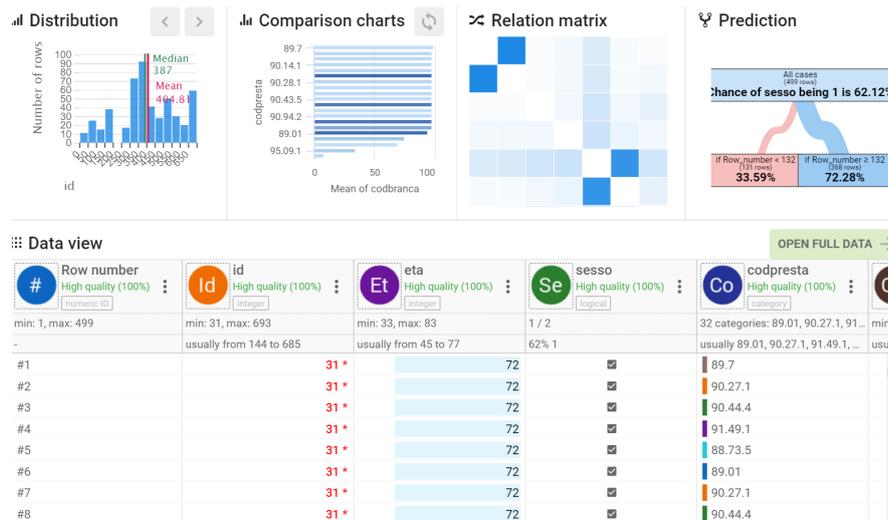


Figure 2.3: General overview of the functionality offered by AnswerMiner

AnswerMiner.

On the home page, we are required to load a dataset in a format like csv or xls. After the uploading we are shown a summary overview of the dataset (Figure 2.3), with a clickable menu on the left side of the page, from which we can access in more detail the different sections. These sections are five and they include:

- ‘data view’, a spreadsheet visualization of the dataset in which we can have an overview of the data (distributions, most common values, outliers)
- ‘relations’, where we can see the correlation degree between each pair of attributes. These correlations are shown both as a map, a matrix and a table
- ‘suggested charts’, where the user can select one or more attributes of the dataset and visualize it through lots of different charts and visualization methods
- ‘prediction tree’, where the user is asked to select a target and one or more predictors to generate a decision tree which aims to show which attributes are most significant for the target prediction
- ‘canvas’, a page where the user can find all the charts he had previously saved

2.3 JetPack Data

JetPack Data is a very simple and intuitive data visualization and data analysis platform. You must pay a monthly fee to use this application, but if you respect the constraint of not loading more than two files per month and if the files uploaded have a size lower than 25Mb, so you can use this application for free. Respecting these constraints we then loaded our dataset to see what analysis this application allowed us to do on it. After loading the dataset we are redirected on a page entitled 'suggested charts', which presents a series of different charts describing our data. Each of these graphs represents couple of different attributes and these initial attributes are chosen randomly or by some algorithm that is not described to us. At the top of this page there is a horizontal menu that allows us to access the other sections of this application. These sections are mainly three:

- 'Build my graphs', in this section the user can choose between seven different types of charts and he can analyze pairs of attributes (putting one on the X-axis and the other on the Y-axis) after choosing the type of aggregation to be used (count, average, min, max)
- 'Dashboards', in this section the user can find all the charts he had previously saved
- 'File summary', a section where the user can find very general information about the file uploaded (file type, number of rows/columns, type fields) and he can see the top fifty rows of this dataset

2.4 DataPine

DataPine is a data visualization tool very similar to JetPack Data. In fact it offers almost the same features described in the previous case. The only real difference with the other application is the graphical interface. Also DataPine has a fixed monthly fee to pay to be used, but in this case there is no a free version of the application, so in order to use it we had to make a registration and we were able to take advantage

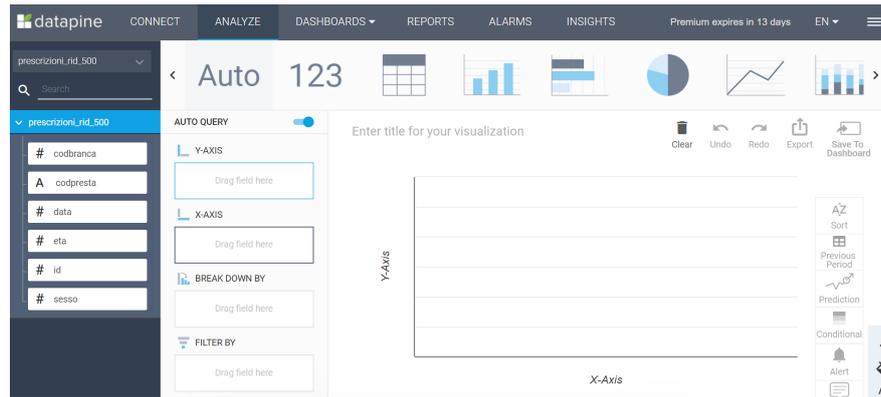


Figure 2.4: Layout of the Datapine analysis page

of the free 14-day trial period that the site offered.

After uploading our dataset, we are shown his first ten rows in a tabular form and we are given the opportunity to eliminate some attributes before starting our analysis. The analysis is of the same type as that offered by JetPack Data.

At the top of the page (shown in Figure 2.4) there is a sliding menu where we can choose the type of chart (column chart, bar chart, pie chart, table, scatter plot, ecc) and after choosing the one that best suits his needs, the user must select a field to put on the X-axis, a field to put on the Y-axis and the type of aggregation (sum, count, average). Also in this case all the charts produced can be saved on a dashboard.

2.5 Dive

Dive is a data exploration tool built at the MIT Media Lab in 2018 which allows non-expert users to analyze their data without writing code [3]. In general, existing tools that are able to do data exploration are tools that require a great amount of work due to their close dependence on analysts, who have to manually specify queries using code. What Dive aims to do is to significantly reduce this amount of work, making data analysis available to all types of users, thanks to its user-friendly interface and its simplicity. Let's see in detail how Dive analyzes the data.

To begin, the user has to create a project, within which he uploads his dataset (the dataset must not have more than 1000 rows). After the upload we are now redirected on a new page; on the left side of this page there is a vertical menu containing four

sections, representing the four stages of a data exploration workflow. These stages are shown in Figure 2.5. They are:

- ‘Datasets’, where there is a spreadsheet visualization of the data. Each field is labeled as nominal, ordinal or continuous. For each kind of field it is reported the number of null values, in particular for a nominal field it is reported also some information like the number of unique values and the frequency of each value, while for an ordinal and continuous field some statistics are shown.
- ‘Visualize’, it’s the visualization mode, where the user can see some charts about his data. By default, if the user doesn’t select any attributes, the application returns univariate descriptive visualizations of each attribute. But the user can also specify one or more fields with a click, focusing the charts on them.
- ‘Analysis’, it is the statistic mode where the user can build aggregation tables, correlation matrices relating each numeric field, running ANOVAs to detect differences between groups and conducting simple regressions.
- ‘Stories’, this is a Compose page where a user can find all the visualizations and statistical analysis result that he had saved before.

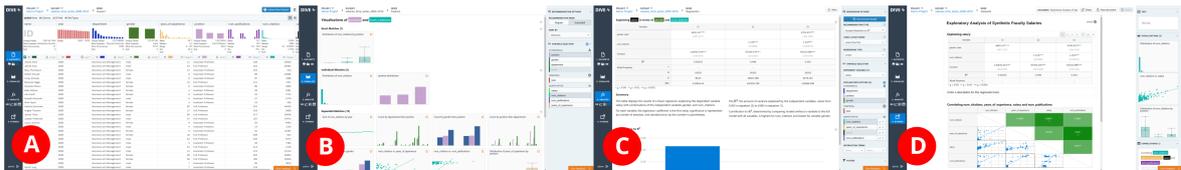


Figure 2.5: *Four stages of a data exploration workflow in DIVE: (A) Datasets, (B) Visualize, (C) Analysis, and (D) Stories*

2.6 NCSS

NCSS is a software that includes over 250 statistical and plot procedures. We used the 30-day free trial to observe the operation of this software and we have seen that the features offered by this software are much more numerous than those offered by

the applications described above. Once our dataset is loaded, we can get information on it by clicking on the top bar on the buttons ‘Analysis’ or ‘Graphics’. All the features offered by these buttons are shown in Figure 2.6.

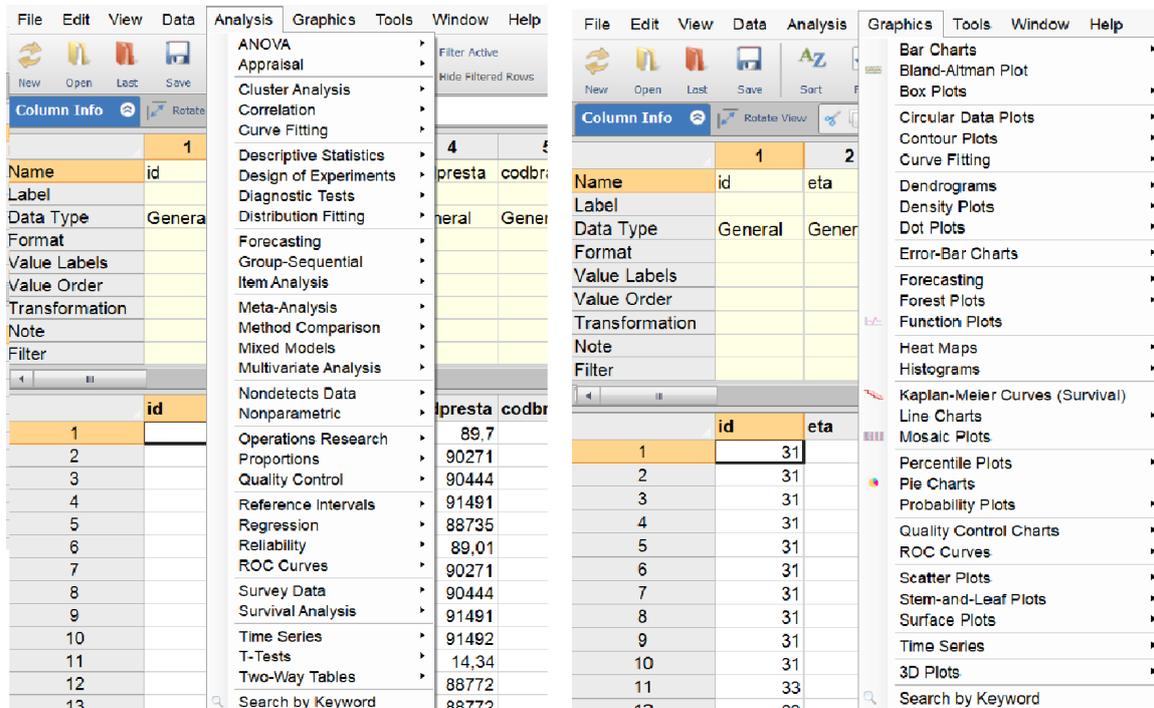


Figure 2.6: Procedures offered by NCSS analysis

Of all these features, we have given particular importance to the section ‘Cluster Analysis’, because it is an analysis that we have decided to implement in our framework and we never found it in the previous applications described. In this section the user can choose between ‘Fuzzy clustering’, ‘Hierarchical clustering’, ‘K-means clustering’, ‘Medoid partitioning’ and ‘Regression Clustering’. We chose a classic ‘K-means clustering’ to see how the results of this analysis were shown. First of all we must set the cluster variables and we must select the number of clusters we want to obtain. A limitation in these choices is that we can select only numeric fields as cluster variables; a text field cannot be used as a cluster variable, because the one hot encoder or other similar transformations are not done in this analysis. After setting the various parameters with appropriate values and clicking on the run button, the results are shown in two ways:

- a textual report, in which for each element the distance from each cluster is indicated, and consequently the cluster to which it belongs. In this report is also indicated the number of elements of each clusters and the mean and the standard deviation of each variable within each cluster
- scatter plots, where a different color is used for each cluster. A limitation of this representation is that all these scatter plots are two-dimensional and each of them represents a pair of variables. So if there are n cluster variables, there will be a total of $\binom{n}{2}$ scatter plots.

2.7 ML Sandbox

ML Sandbox is another online free tool available at <https://intelligentonlinetools.com/cgi-bin/analytics/ml.cgi> that allows us to do the cluster analysis. It is a very poor-graphically tool and it's not very user friendly: for example the user can't upload a dataset, but he has to cut and paste it without column names and with only a white space between each value (not comma or other signs) and a new line character between each row. Anyway, since there are very few online tools that allow any user to perform cluster analysis for free, we decided to test it and see its results.

This tool proposes several clustering algorithms and for each of them the user can select the kind of data normalization (no-scaling, StandardScaler, MinMaxScaler). If the dataset to be analyzed contains text field, these algorithms don't work, because there is no technique to transform text values into numeric values in this tool (like the OneHot Encoder).

The most important of these clustering algorithms are:

- K-Means clustering. The user must set as a parameter the number of clusters that he wants to obtain and then he has to click on the run button. As a result there is a vector in which for each element of the initial dataset there is a cluster Id that indicates which cluster it belongs to. For each cluster the coordinates of its centroid are also indicated. Finally, there is a 2d scatter plot in which each cluster is represented with a different color. The scatter plot axes correspond

to the first two columns of the initial dataset (there is no SVD or PCA)

- DBSCAN. The user can still select the number of clusters he wants to obtain, but obviously this parameter does not make sense in this algorithm. The parameter necessary for this algorithm (MinPts and eps) are chosen automatically (we don't know according to which criterion) and can't be modified by the user. As a result there is simply a vector of cluster Id that tells us each element which cluster it belongs to
- Hierarchical Clustering. In this case the choice of the number of clusters by the user is useless, because as a result there is only a vertical dendrogram. The user cannot choose the kind of aggregation (min, max, average) and we don't know which of these was chosen

Other types of cluster algorithms that this tool offers are: text clustering, affinity propagation clustering, BIRCH clustering and CURE clustering.

Chapter 3

Automated Data Exploration

In this chapter we will present our proposed framework for automatically analyzing a dataset, using data mining techniques, and we will explain the theory behind the various techniques and algorithms we used.

In general, the knowledge extraction process includes basic building blocks, which are selection, preprocessing, transformation, extraction and visualization (these blocks are shown in Figure 3.1). But this process is an iterative process that requires great expertise from the data scientist, because for each of these building blocks there are so many algorithms that must be properly configured by the analyst and must be chosen based on the type of data we are analyzing.

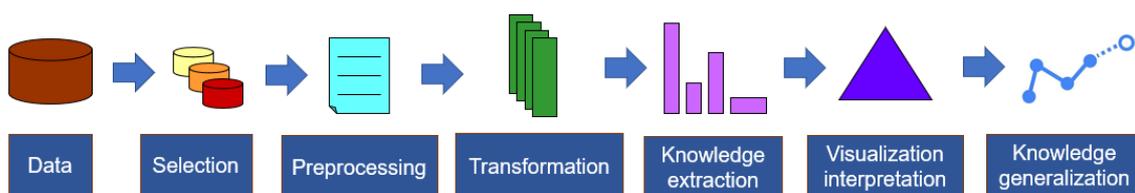


Figure 3.1: *Traditional Knowledge Extraction Process*

So, the aim of this thesis is to try to automate the process, minimizing the contribution of the data scientist, with the final objective to democratize data science. For this purpose we have proposed an automatic knowledge extraction workflow. The workflow proposed by us includes the following steps: data type identification, data characterization, outlier detection, cluster analysis, classification, data trans-

formation and visualization. These steps can be summarized as shown in the figure 3.2.

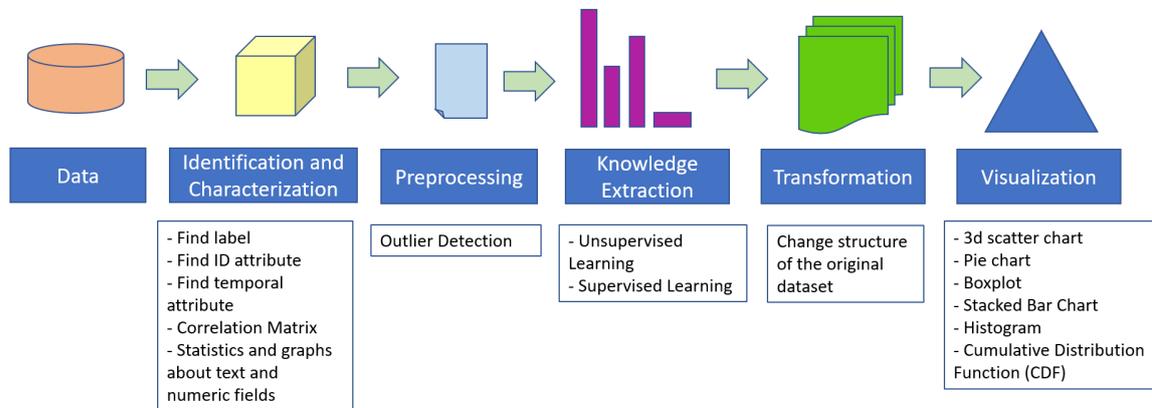


Figure 3.2: Automated Workflow proposed by the framework

3.1 Data type identification

In this first step, we find the general characteristics concerning our dataset. After having divided the various fields into numeric fields and text fields, we look if there is any particular attribute, like a label, an ID, or a temporal attribute. Label and temporal attribute searches are done exclusively at the semantic level. The first comparing the names of the various attributes with words like 'class', 'type', 'label', 'species', 'outcome' (both in uppercase and lowercase). The second comparing the names of the attributes with words like 'data', 'day', 'month', 'year'.

For the ID search, initially a semantic comparison is always made, looking for attributes that contain the word 'ID' in its name. If this first search is successful, then we check if the number of distinct values of that attribute is equal to the total number of rows of the dataset. If the two numbers are the same, then we found an id attribute.

3.2 Data characterization

In this second step of the workflow interesting information on the data is provided. This information can be either in the form of statistics or in the form of graphs.

For a textual attribute, the reported statistics are:

- Number of distinct values
- Most frequent value and number of times it appears in the dataset
- Number of null values

While for a numerical attribute the reported statistics are:

- Mean (sum of all values, divided by the total number of rows)
- Standard Deviation, obtained with the formula:

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - M)^2}$$

where N is the number of values present in the attribute, x_i is the single element and M is the mean value of the attribute

- Range within which the attribute values are included
- Number of null values

Moreover, for the numerical attributes the correlation matrix is calculated [4], that is a matrix showing the Pearson Correlation Coefficient between each pair of numerical attributes. This coefficient is an index between -1 and 1 which expresses a possible linearity relationship between the variables. Given two variables, the Pearson correlation index is defined as their covariance divided by the product of the standard deviations of the two variables. If this index is close to 1, it means that the two variables are positively correlated: increasing one variable, the other will also increase. If, on the other hand, the index is close to -1, it means that the two variables are negatively correlated: decreasing one variable, the other will increase. Obviously the diagonal of this matrix is formed by all 1, because there is the coefficient between equal attributes.

In addition to the correlation matrix, the following graphs are also obtained for numerical attributes:

- Boxplot, a graph that intuitively describes the characteristics of the distribution (see paragraph 3.3)
- Cumulative Distribution Function (CDF), in which the x-axis represents the number of occurrences of a given value, while on the y-axis all the distinct values of the attribute are represented. If the curve were a straight line, it means that all values are repeated only once; while long horizontal sections in the curve (parallel to the x-axis) indicate that there are more occurrences for single values
- Histogram, a bar chart where the x-axis contains all the values between the minimum value and the maximum value of the attribute. This range is divided into 10 parts of equal size and for each of these parts a vertical bar is built that tells the user how many values of the attribute are contained in that range

3.3 Outlier detection

The term 'outlier' means an anomalous value within the dataset, clearly distant from the other available data. Since these anomalous values can influence many indicators, such as the average or the standard deviation, before doing our analysis we decided to identify and eliminate them. In our framework two different techniques are used to do the outlier detection:

- ANOVA and Boxplots analysis (if a label is present)
- DBSCAN (if there is no label)

ANOVA and Boxplots analysis

In this case, first of all we searched for the most significant attributes of our dataset using a hypothesis test. A hypothesis test is a statistical test used to understand if a certain condition can be applied to an entire population, analyzing only a small sample of this. The test result tells us whether to believe or reject the null hypothesis. In our case we analyze each attribute of the dataset (excluding the label found)

and for each of them the hypothesis to be tested will be: 'This attribute has no importance for the response label'. Doing so we want to try to find those attributes that give a greater contribution than the others in choosing the final label. To find these attributes we perform an ANOVA [5] univariate test (a kind of hypothesis test) on each of them and we assign this attribute a p-value. P-values are numbers between 0 and 1 that represent the probability that the data analyzed are due to chance; this means that the lower the value found, the better it is to reject the null hypothesis, which in our case means that the analyzed attribute can be considered meaningful. To determine which attributes to consider as significant and which are not, we used a threshold value for the p-value of 0.05 (anything less than 0.05 is considered significant).

After having found the most significant attributes, for each of them we find the eventual outliers analyzing their boxplot representation. The boxplot is a graph used to represent a distribution [6]. This graph is based on 5 numbers, these numbers are:

- *minimum*, the minimum observed value
- *first quartile (Q1)*, the minimum observed value such that at least 25% of data is less than this or equal to this.
- *median*, the minimum observed value such that at least 50% of the data is less than this or equal to this.
- *third quartile (Q3)*, the minimum observed value such that at least 75% of data is less than this or equal to this.
- *maximum*, the maximum observed value

From these data we can find the interquartile range (IQR), defined as the first quartile subtracted from the third quartile: $IQR = Q3 - Q1$. After obtaining this value we have therefore set the minimum value like $MIN = Q1 - 1.5 * IQR$ and the maximum value like $MAX = Q3 + 1.5 * IQR$. These two numbers that we have set are our thresholds that allow us to determine if a value is to be considered outlier or not. In fact all the values less than the minimum or greater than the maximum are labeled as

outliers.

This boxplot analysis (shown in Figure 3.3) for the outlier detection is performed

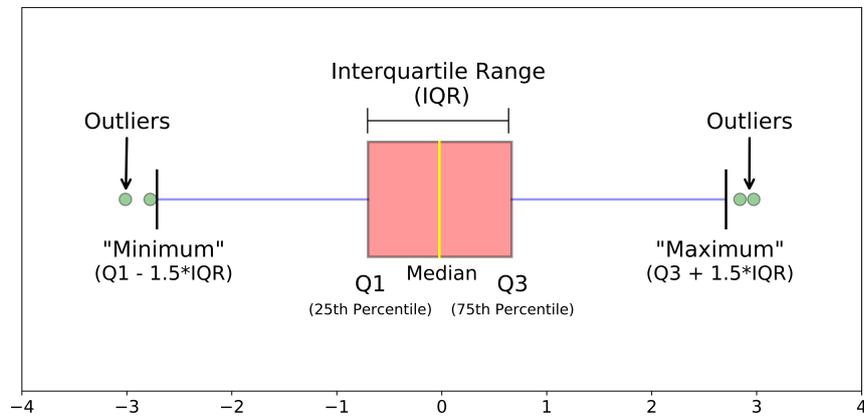


Figure 3.3: *Different parts of a boxplot [6]*

once for each attributes that have been recognized as 'meaningful' by the ANOVA test and it returns a set containing all the values recognized as outliers. At the end of all, these sets are joined together, forming a single set that contains all the outliers of our dataset.

DBSCAN for outlier detection

DBSCAN is a density-based clustering algorithm that classifies points into core points and border points. Any point that is not recognized as either core point or border point is labeled as noise (outlier). However, before using this dbscan algorithm, we transformed the textual attributes into numerical attributes and we standardized our data (techniques described in the next section).

How the DBSCAN algorithm works and how its parameters are chosen, is explained in depth in the next paragraph.

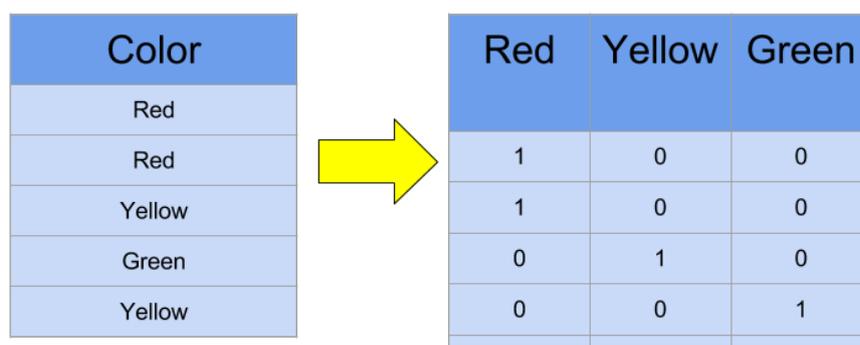
3.4 Self-tuning cluster analysis

In this step three clustering algorithms are performed on our data: k-means [7], dbscan [7] and hierarchical clustering [7], applied with a self tuning strategy (automated parameter estimation) to automatically configure specific input parameters.

Each of these algorithms requires parameters that significantly affect their performance. Our framework then tries to find these parameters automatically, so as to maximize the algorithm's performance, minimizing user effort.

But before applying our clustering algorithms, we had to make some transformations on our data. First of all, since the clustering algorithms we will use are based on distances between the various points, all the attributes of our dataset must be numeric fields. To do this we must transform text fields into numeric ones. To transform the data into numerical values we therefore decided to use the One Hot Encoder() function, provided by the python sklearn library, which operates as follows [8]: considering for example a 'color' attribute, which has as possible values [green, red, yellow], the One Hot Encoder() function splits the 'color' column into three different columns called 'Red', 'Yellow', 'Green' and assigns a 0 or a 1 as the value of these columns, depending on the textual value in the original dataset. At the end of this operation our dataset will be composed entirely of numerical values. The negative aspect is that this transformation leads to an increase in the number of columns of the dataset. A graphic example of how this technique works is shown in figure 3.4.

Furthermore, as the various clustering and classification algorithms we are going



Color			
Red			
Red	1	0	0
Yellow	1	0	0
Green	0	1	0
Yellow	0	0	1

Figure 3.4: Example of how the One Hot Encoding Works [9]

to apply work better when the attributes are on a relatively similar scale, after transforming our data into only numeric values, each feature of the dataset has been standardized. After standardization, each attribute will have mean values equal to zero and variance equal to 1. To achieve this, the average of an attribute is subtracted from each value of that attribute, and this result is divided by the standard

deviation of the attribute:

$$z_i = \frac{x_i - \mu}{\sigma}$$

Now let's see in detail the clustering algorithms we used.

3.4.1 K-Means Clustering

K-Means [7] is an unsupervised learning algorithm that finds a fixed k number of clusters in a data set. This is an iterative algorithm, because it repeatedly executes some steps. These steps are:

- Step1: it chooses a number k of clusters in which to divide the data and chooses k randomly positioned initial centroids
- Step2: in this phase the algorithm analyzes each data point and assigns it to the nearest centroid
- Step3: now the position of the centroids is recalculated, as adding (or removing) new points the cluster will have changed. The new value of a centroid will be the average of all the data points that have been assigned to the cluster.

Steps two and three are repeated until the centroids no longer change (a point of convergence has been reached). In this case the stop condition has been reached.

This condition is usually represented by one of the following options:

- no data points change cluster
- the sum of the distances is reduced to a minimum
- a maximum number of iterations is reached (in our case this number is set to 300 by default)

Automated parameter estimation

The choice of the k parameter is fundamental in the k -means algorithm. If we know a priori the number of clusters we want to get, we obviously set k to that number;

but if we don't want to find a particular number of clusters, but instead we simply want to get the best solution among those possible, the strategy to adopt is the following.

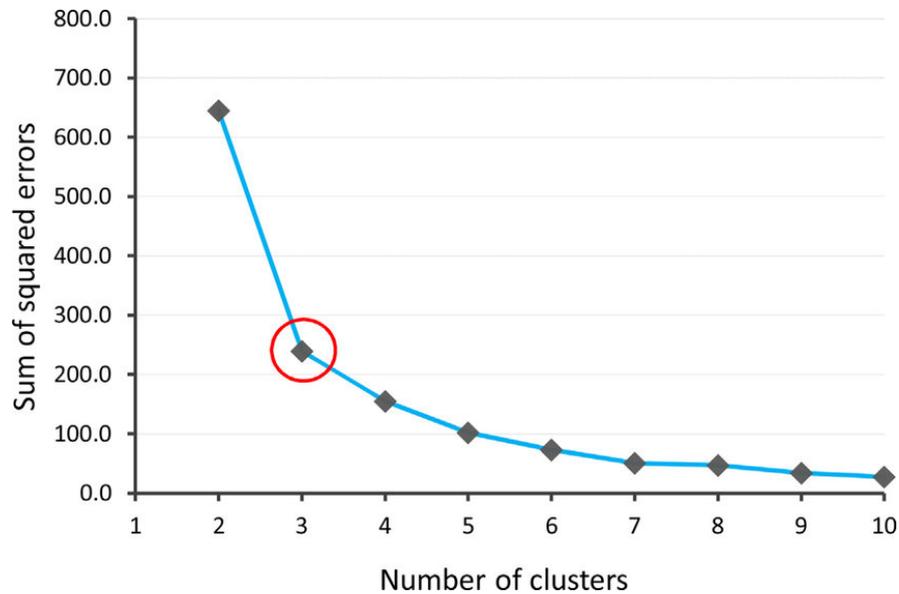


Figure 3.5: Result of the elbow method for choosing the best number of clusters [10]

To look for the best solution of our algorithm, we calculated the k-means for each value of k in the interval $[1,10]$ (we have limited the maximum number of clusters to 10) and for each of these solutions we have calculated the SSE (Sum of Squared Error), which is the sum between the distances of each point and its nearest centroid. This sum decreases when k increases and vice versa it increases when k decreases, because an increase in the number of clusters is related to smaller distances. The goal of this process is to find the point where the increase in k will cause a very small decrease of the SSE, while the decrease in k will sharply increase the SSE. If we plot our results in decreasing order of SSE, in a graph where on the x-axis there is k and on the y-axis there is the SSE score, the point we want to find is called the elbow point of the curve. To calculate this elbow point automatically [11], we have drawn a straight line between the first point and the last point of our curve and we looked for the point of the curve with the greatest distance from the line.

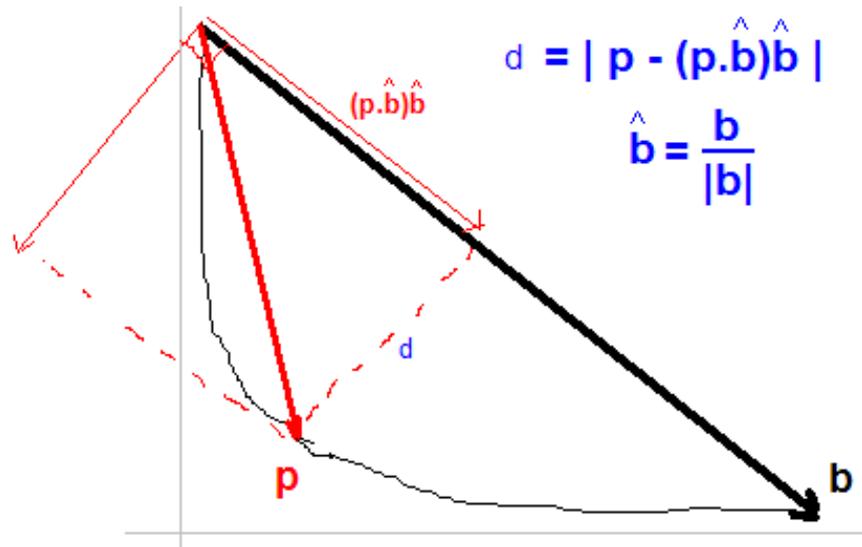


Figure 3.6: Find the elbow of a curve, looking for the point with the greatest distance from the line b (between the first and the last point of the curve) [12]

This point is our elbow point and its coordinate on the x -axis is the value of k we were looking for.

3.4.2 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [7] is a density-based clustering algorithm, because it connects regions of points with sufficiently high density. This algorithm is based on two parameters: a radius eps and a minimum number of points $minPoints$. Starting from these two numbers the data points are classified in the following way:

- a *Core Point* is a point that, in its neighborhood of radius eps , has a number of points greater than or equal to $minPoints$
- a *Border Point* is a point that it is not a core point, but it is in the neighborhood of a core point
- a *Noise Point* is any point that is not classified as a core point, nor as a border point

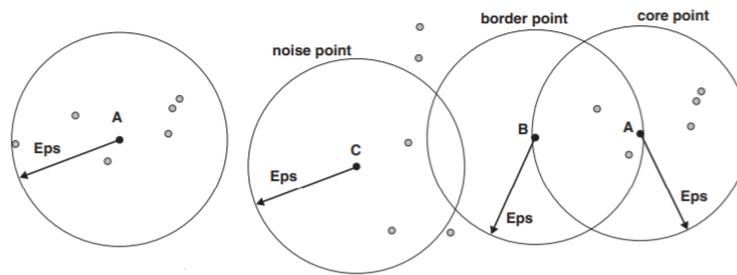


Figure 3.7: Example of based density on the left. Core, border and noise points on the right [7]

From the definitions above, three more definitions can be derived, which are fundamental to understand well the concept of cluster in the DBSCAN algorithm:

- a point a is *Directly Density Reachable* from a point b if b is a core point and a is in a neighborhood of radius ϵ of b . This notion is not symmetric
- a point a is *Density Reachable* from b , if it exists a sequence of points p_1, \dots, p_n , where $p_1 = a$ and $p_n = b$ and each point p_{i+1} is directly density reachable from p_i . This notion is not symmetric
- two points a and b are *Density Connected* if there is a point x such that both a and x , both b and x are density-reachable. This notion is symmetric

These notions are very important to understand the algorithm, in fact all the points inside the same cluster are mutually density connected. Let's now analyze the main steps of this algorithm:

- First of all we choose an arbitrary point that has not yet been visited
- Then we calculate the number of points in its ϵ -neighborhood and if this number is greater than or equal to minPoints , then the point is labeled as core point and a new initial cluster is created; otherwise the point is initially labeled as a noise point (but it can be later be in a ϵ -neighborhood of another point and become part of a cluster)
- If the point is a core point, then all the points in its neighborhood are added to its cluster label. If there are other core points between these points, the points

in their eps-neighborhood are also added to the cluster. And so on until the cluster is completed

- The previous steps are repeated until all the points are visited

Automated parameter estimation

Eps and minPoints are two fundamental values for the algorithm to work correctly. For example if we choose a value of minPoints that is too low, new clusters will be built between noise points. With an eps value too low, most of the points will be considered noise, because there are not enough points in its neighborhood to be a core points. On the other hand, with an eps value too high, the different clusters will merge together and most of the points will be in the same cluster.

With these examples above it is evident that the two parameters should be chosen appropriately. There is no universally correct method to do this, as the situation changes according to the dataset analyzed. What is shown below is our method for automatically choosing the two parameters, without the user having to enter them manually.

Eps

Indicating with d the distance of a point p from its k -th nearest neighbor, within a distance d from p there will be $k+1$ points. After setting the value of k , we define a k -dist function that assigns to each point of the dataset the distance from its k -th nearest neighbor. Subsequently we order these distances in descending order, obtaining a graph called 'sorted k -dist graph'. In this graph, choosing an arbitrary point p and assuming that k is equal to minPoints and that k -dist(p) is equal to eps, all points with less or equal k -dist will be core points. Therefore, having chosen a threshold point on the graph (which corresponds to a value of k -dist and therefore of Eps), all the points preceding it will be labeled as noise, while the subsequent points will be assigned to clusters. This point is automatically selected by our application looking for the elbow of the k -dist graph, with the same technique described in section 3.3.1.

MinPoints

We have just explained how to find eps starting from a k-dist graph, but how do we choose the k-dist to use? In particular, how do we choose the value of k, which is equivalent to our minPoints?

In the technique we used to solve this problem we have limited the values of k in a range between 2 and 50. For each value of k in this range we have calculated the corresponding k-dist graph and, starting with k=2 and increasing this value by 1 each time, we have seen the differences with the (k+1)-dist graph. To compare two curves we used the mean absolute percentage error (MAPE), whose formula is given below:

$$M = \frac{100}{n} \sum_{t=1}^n \frac{|A_t - F_t|}{A_t}$$

where A_t is a point in the k-dist graph, F_t is a point in the (k+1)-dist graph and n is the number of points in our dataset.

This formula returns a percentage value and, if this value is less than 3 (the difference between the two curves is small), we stop the algorithm and we choose the current k as minPoints to use in our DBSCAN. This threshold value (3) was chosen empirically, applying this technique to different datasets.

3.4.3 Hierarchical Clustering

Hierarchical clustering [7] is a clustering approach that aims to build a cluster hierarchy. Hierarchical clustering strategies are typically of two types:

- Agglomerative, in which at the beginning each element forms its own cluster, and these clusters are then gradually grouped together by two
- Divisive, where all the points at the beginning are in the same cluster, and this cluster is then recursively divided into sub-clusters

In our application we have only adopted the agglomerative strategy. The basic algorithm of this technique is the following:

- STEP1, the distance matrix is calculated, which contains the Euclidean distance between each point and all the other points
- STEP2, each point is assigned to a different cluster (n points \rightarrow n clusters)
- STEP3, the two closest clusters are joined together in a single cluster
- STEP4, the distance matrix is updated

STEP3 and STEP4 are repeated until a single cluster remains.

The result of a Hierarchical clustering is represented graphically in a dendrogram. A Dendrogram is a tree-like diagram, useful to understand how the various clusters have been grouped together. The horizontal axis of this diagram represents the objects and clusters, while the vertical axis of the dendrogram represents the distance between clusters. But how can we calculate the similarity between two clusters, to decide which of these join together? There are several approaches used to calculate the similarity between clusters, the most important are the following:

- MIN, where the similarity between two clusters is equal to the minimum distance between two points P_i and P_j , where P_i belongs to Cluster1 and P_j belongs to Cluster2

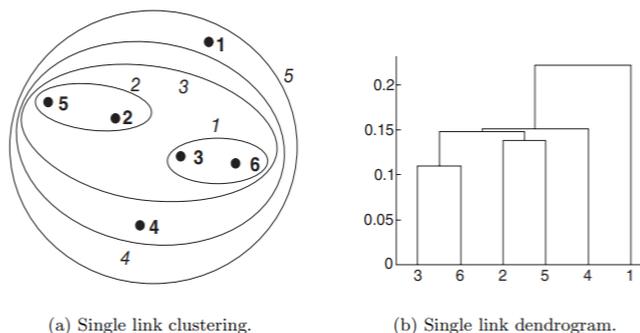


Figure 3.8: Hierarchical Clustering using 'min' as a criterion of similarity between two clusters [7]

- MAX, where the similarity between two clusters is equal to the maximum distance between two points P_i and P_j , where P_i belongs to Cluster1 and P_j belongs to Cluster2. This approach is the exact opposite of the previous one

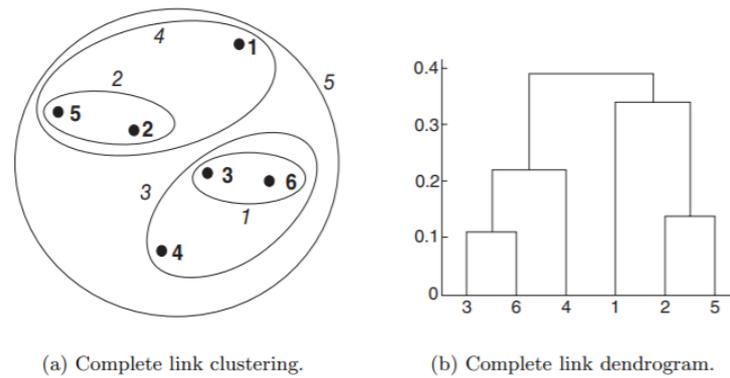


Figure 3.9: Hierarchical Clustering using 'max' as a criterion of similarity between two clusters [7]

- Group Average, where the similarity between two clusters is the average of all the possible distances between two points P_i and P_j , where P_i belongs to Cluster1 and P_j belongs to Cluster2

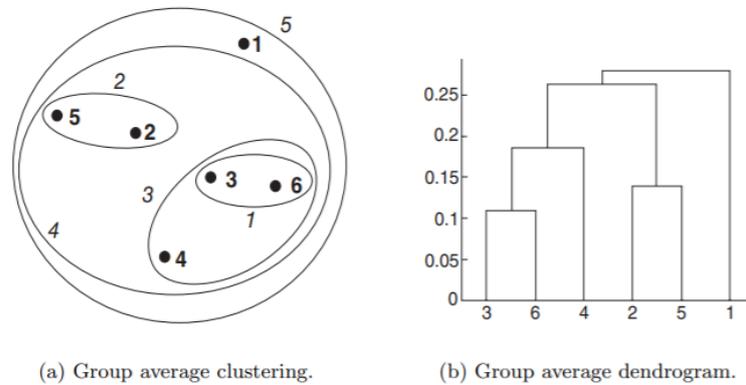


Figure 3.10: Hierarchical Clustering using 'average' as a criterion of similarity between two clusters [7]

- Ward's Method, where the approach is the same used in the Goup Average, but in this case the sum of the squares of the distances between P_i and P_j is calculated

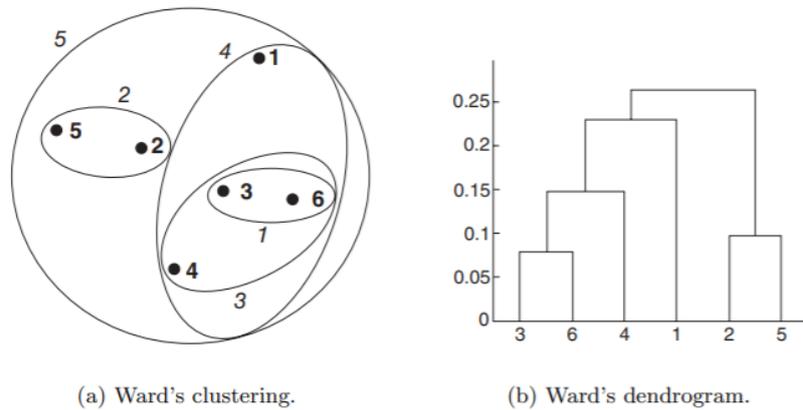


Figure 3.11: Hierarchical Clustering using 'ward' as a criterion of similarity between two clusters [7]

Automated parameter estimation (Silhouette)

Silhouette is a technique used to evaluate how good a clustering solution is [13]. This technique measures how similar an object is to its own cluster and how far it is from other clusters. For each point i , the Silhouette index $s(i)$, between 0 and 1, is calculated as follows:

$$s(i) = \frac{b(i) - a(i)}{\max[a(i), b(i)]}$$

where $a(i)$ is the average distance between the point i and all the other points that belong to its own cluster and $b(i)$ is the minimum average distance between point i and all the other clusters, to whom i does not belong.

The value $s(i)$ tells us how well the data has been clustered, in particular:

- $s(i) \sim 1$ means that the point i has been correctly assigned to the cluster
- $s(i) \sim -1$ means that the point i was not assigned to the right cluster
- $s(i) \sim 0$ means that the point is not clearly assignable to one of the clusters

To evaluate the goodness of all our clustering algorithm, we use the Average Silhouette which is the average of all the silhouettes $s(i)$ in our dataset.

In particular, to automatically select the number of clusters to be shown initially to the user, we have done Hierarchical clustering ten times, the first with 2 clusters,

the second with 3 clusters and so on until we have 11 clusters. For each of these algorithms we have calculated the value of Average Silhouette and we have chosen the algorithm with this highest value.

3.5 Classification

The framework uses classification techniques in two different contexts. In a first case, after some clustering algorithm has been applied and a label has been given to data, a decision tree is built on the new dataset, which shows the user what attribute values make a data come to be labeled in a certain way (this classification technique is applied to all datasets after clustering analysis).

In a second case, different classification techniques are applied only if the loaded dataset contains a label and the various accuracy of these algorithms are compared with each other. The algorithms used are the following:

Decision Tree

A Decision Tree [14] is a structure that allows us to understand which are the attributes that have a greater relevance in the assignment of the label outcome. Each node of this tree verifies a condition (test) on a given attribute and each branch descending from this node represents one of the possible test results (one of the possible values of the attribute), while each leaf node represents the label outcome. The first node in a decision tree is called root node and it is important that the first test done in this node is done on the attribute that best classifies the data, so that the classes are separated as much as possible from each other after this first test. To determine the attribute that best classifies the data we use Gini index, which measures the impurity of the dataset for a given node t .

The Gini Index for a node t is:

$$GINI(t) = 1 - \sum_c [p(j|t)]^2$$

where $p(j|t)$ is the relative frequency of class c at node t . This index takes the value 0 in the best case in which all the data belong to the same class and it grows when

the data are distributed in a more heterogeneous way among the classes.

The algorithm uses this index to determine how good a split is, looking at how mixed the classes are within the two groups created by the split.

Support Vector Machine

The Support Vector Machine (SVM) is a supervised machine learning algorithm. The objective of this algorithm is, given a set of data belonging to N different classes, to find a hyperplane in an N-dimensional space that divides the classes in the best possible way.

With the term 'support vectors', we indicate those points that have a shorter distance from the hyperplane. The distance between the support vectors of two different classes is called margin. Since given a set of data belonging to two classes, there could be infinite hyperplanes that divide the data correctly, the SVM proposes to find, among all the possible solutions, the optimal one, that is the one with the largest margin.

K-Nearest Neighbor

The KNN algorithm (K-Nearest Neighbor) is a Machine Learning algorithm that is based on the concept of classifying an unknown sample, considering the class of k nearest samples of the training set. The new sample will be assigned to the class to which most of the nearest k samples belong. The choice of k is therefore very important for the sample to be assigned to the correct class. If k is too small, the classification may be sensitive to noise, if k is too large the classification can be computationally expensive and the neighborhood may include samples belonging to other classes.

Naive Bayes Classifier

The Naive Bayes Classifier is a classifier based on the Bayes theorem. With this theorem we can get the probability that A happens, given B:

$$P(A|B) = \frac{P(A|B)P(A)}{P(B)}$$

Suppose in our dataset we have n features (X_1, X_2, \dots, X_n) and a label y that can have m distinct values. The probability that a generic line from our dataset (of which we know the n features) will have a label Y , will be:

$$P(Y|X_1, \dots, X_n) = \frac{P(X_1|Y)P(X_2|Y)\dots P(X_n|Y)P(Y)}{P(X_1)P(X_2)\dots P(X_n)}$$

This probability is calculated m times, each time with a different Y value (Y_1, \dots, Y_m) . Of all these calculated probabilities the highest one is taken and the element of our dataset is labeled with the label (the value of y) associated with that probability.

3.6 Data Transformation

The framework offers the possibility of transforming the loaded dataset, modifying its structure. The data is then reported in a different way, to try to extract information that are too hidden or absent in the original structure. How the dataset is transformed and when these transformations are applied, is explained in detail in chapter 4.4.

3.7 Visualization

In this framework, the most used way to graphically represent the distribution of the data (outlier detection, cluster analysis, etc.), is a clickable 3d scatter chart, easily understood by any user. But to get this representation, all the attributes of our dataset must be summarized in only three attributes (one per Cartesian axis). To do this we used the Principal Component Analysis (PCA).

PCA [15] is a non-supervised learning algorithm that reduces the dimensionality (number of features) of a dataset, while maintaining as much information as possible about the data. This algorithm reduces the dimensionality by identifying a new set of features called components, which are composite of the original characteristics unrelated to each other. The first component represents the largest possible variance in the data, the second component the second largest variance and is orthogonal to the previous component, and so on.

To implement the PC we act as follows:

- we have standardized the data: after the standardization, all the attributes of our dataset will have mean equal to 0 and variance equal to 1
- we calculated the covariance matrix, which is a symmetric matrix that represents the variation of each variable with respect to the others
- we calculated the eigenvalues and the respective eigenvectors of the matrix
- the eigenvectors are nothing more than the new components we were looking for and the larger an eigenvalue is, the more importance the corresponding eigenvector will have
- we ordered the eigenvectors in descending order of importance and we took the first three of these eigenvectors (3 principal components)

Other types of charts that are used in the framework to show the user the data distribution are:

- Histograms (bar charts)
- Boxplots
- Cumulative Distribution Function
- Pie Charts
- Stacked bar charts

Chapter 4

Architecture and Workflow

The data exploration process includes several tasks that are often implemented by different tools. To avoid having to use multiple tools and to simplify the exploration process by the user, we have tried to unify all these tasks in a single automated framework. In our process of data exploration, we have tried to show to the user the information, structures and models hidden in the data, following a logical order and dividing the results into different exploitable and human readable sections. The logical order in which the data is analyzed automatically by our framework is illustrated from the content in figures 4.1 and 4.2:

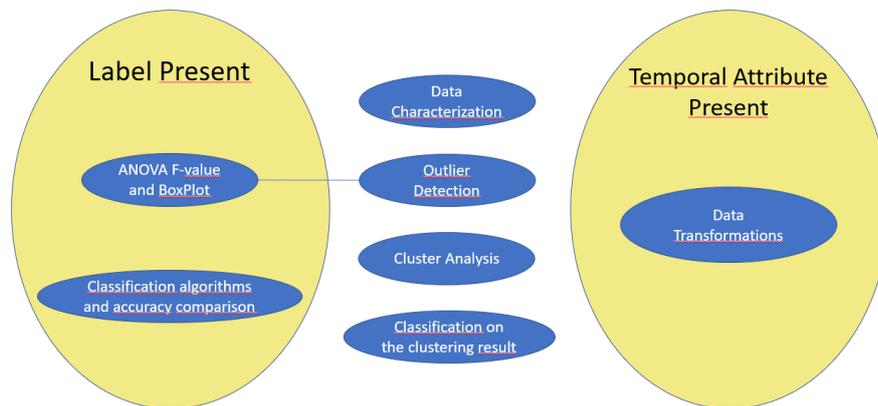


Figure 4.1: Map containing the different data exploration tasks offered by the framework

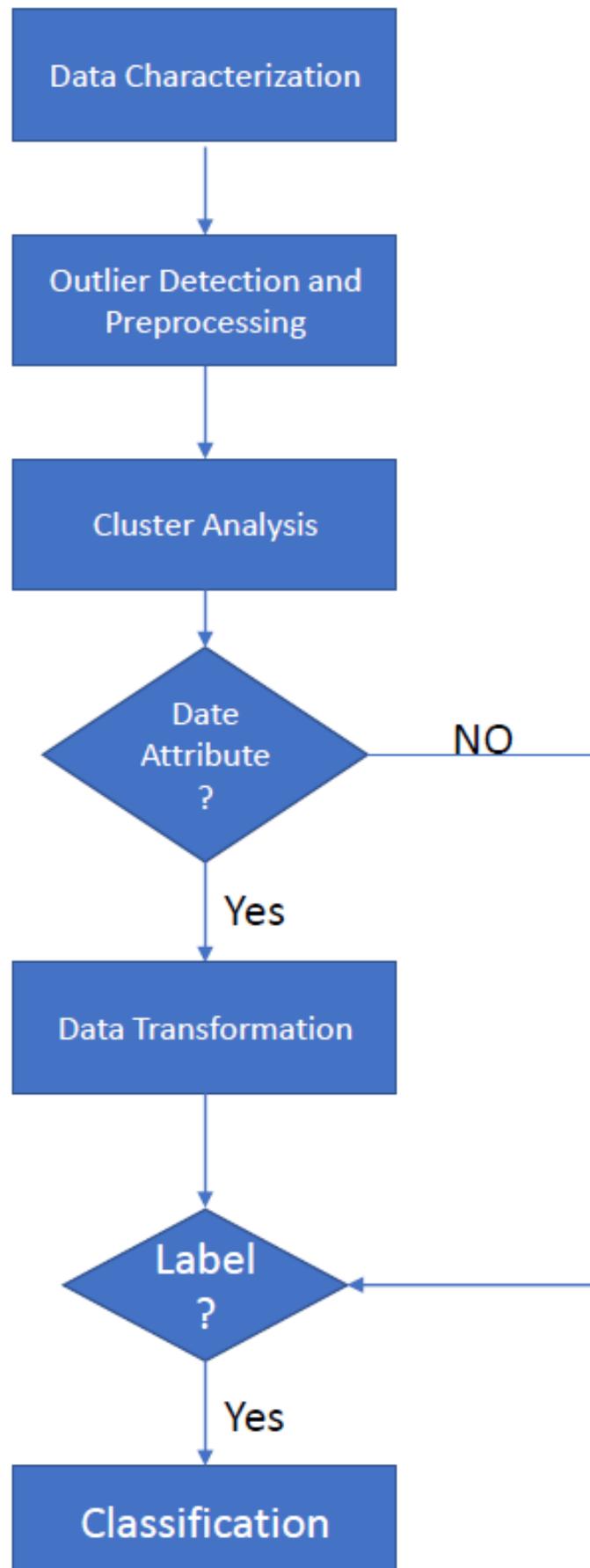


Figure 4.2: WorkFlow of the automated data exploration proposed by the framework

In any view of our application, the user can always see a vertical navigation bar on the left of the page, from which with a simple click he can freely access the various sections. These sections are three and are called 'Data', 'User Exploration' and 'StoryTelling'. The section 'User Exploration' then groups together three other subsections, which are called 'Info', 'Clustering' and 'Data Transformation'. The first page of the application is an upload page, where the user can load his dataset by searching for it manually, or by dragging it inside the dotted rectangle (see figure 4.3). The only accepted formats for datasets are .xls and .csv. If a user tries to upload a file of another format, a red error message is shown and he does not leave the upload page. If the upload is successful, the user is redirected to the 'Data' section, from which the data exploration can begin.



Figure 4.3: *Upload Page*

Now, let's see in detail the various sections in which the user can navigate.

4.1 Data

This first section shows the basic information of the loaded dataset, such as the name, the number of lines and the number of columns. There is also a representa-

tion of the whole dataset in a table form. The user can scroll the dataset vertically, seeing all the elements contained in it. From this representation the user can also obtain some information about the nature of the various attributes, in fact the numeric fields are shown in black, while the text fields are highlighted in blue. Furthermore, if some null value is present in the loaded dataset, the whole line containing that value is highlighted in red. If any rows contain null values, these rows will be deleted before the dataset is analyzed.

4.2 Data Characterization

This section shows the user more detailed information regarding the data he has uploaded and this section is generically named 'Info' in our developed tool. This information is initially represented in the form of statistics. In addition to these there are also several graphs showing the user various information regarding each attribute, in an intuitive and easy understandable way.

First of all, our framework distinguishes textual attributes from numerical ones. If the attribute is a text field, it is reported if it is a categorical attribute or a non-categorical attribute (in our application, an attribute is defined categorical if the number of distinct values for that attribute is less than the 20% of the total number of rows in the dataset). For a text field is also reported what is the number of distinct values, what is the most present value (and how many times it appears) and how many are the null values in that attribute.

Regarding the charts for this type of attributes, for each text field of the categorical type its distribution in a bar graph is represented, that is a histogram that contains a bar for each distinct value of the attribute and that on the y axis shows a scale that indicates the number of occurrences for each value.

If the attribute is a numeric field, first of all there are some statistics that describe the data, such as:

- The average value
- The standard deviation

- The minimum value
- The maximum value
- The number of null values present on it

After these details, a series of graphs are shown. First of all there is a correlation matrix [4], that is a matrix showing the Pearson Correlation Coefficient between each pair of numerical attributes. After the correlation matrix, the following three graphs are then shown for each numerical attribute:

- Boxplot, a graph that intuitively describes the characteristics of the distribution
- Cumulative Distribution Function (CDF), which is a growing curve that helps the user understand the distribution of the attribute
- Histogram, a bar chart formed by 10 bars of the same width (they refer to ranges of equal size), each of which shows the user how many values of the attribute are contained in that range

Obviously if the loaded dataset contains an 'Id' attribute, the graphs on this attribute will be of little interest. For this reason, our application looks for an 'Id' attribute and if it finds it, is eliminated momentarily, so that it does not appear in the correlation matrix and the graphs on this attribute are not shown.

4.3 Exploratory and Unsupervised learning

In this third section of our application, we try to provide the user with a simple and intuitive cluster analysis, representing each result obtained with very user-friendly graphs. The various steps we have implemented in this section are as follows:

- Outlier Detection and Removal
- Cluster Analysis and Parameter Estimation
- Cluster characterization

Outlier detection and removal

First of all the 'outlier' points of our dataset are eliminated, in order to remove as much as possible the noise and to have a cleaner dataset. This 'outlier detection' is shown to the user as an interactive 3d scatter chart which represents all the points of our dataset according to its 3 most significant dimensions (found by applying the PCA [15]), where all the points labeled as 'outliers' are shown in red and the other points are shown in green.

Cluster Analysis and Parameter Estimation

In this section the results of three clustering algorithms are shown: K-Means, DBSCAN and Hierarchical Clustering. The result of each of these algorithms is shown in a 3d scatter chart generated with HighCharts, where the points belonging to the same cluster have the same color.

The first solution shown is that in which the parameters are automatically selected by the framework, but later, the user can also interact with the application, manually entering the value of these parameters and thus being able to view other results. In order to adequately choose these parameters, for each of these algorithms the user is shown a different graph able to help him in the choice. In particular, these graphs are:

- *The elbow graph* for the k-means, thanks to which the user can visually identify the elbow point of the curve and choose the number of clusters accordingly
- *The k-dist graph* for DBSCAN. The value of k (MinPoints) is chosen automatically by the framework, but starting from this value the user can again choose eps by visually evaluating the elbow of the curve
- *The dendrogram*, thanks to which the user can see how the various clusters are associated and how far they are from each other. Evaluating these distances, it is possible to decide at what height to cut the dendrogram and based on the cut to obtain a different number of clusters

Cluster Characterization

After applying a clustering algorithm on the data and assigning a label to each value, the framework shows different results and information regarding this cluster analysis. First a pie chart is shown, which, with the same colors of the scatter chart, tells us how many elements belong to each cluster and the respective percentages. Other information on the result of the clustering algorithm is shown in the following graphs:

- Radar Chart
- Decision Tree
- Stacked bar chart for cluster validity

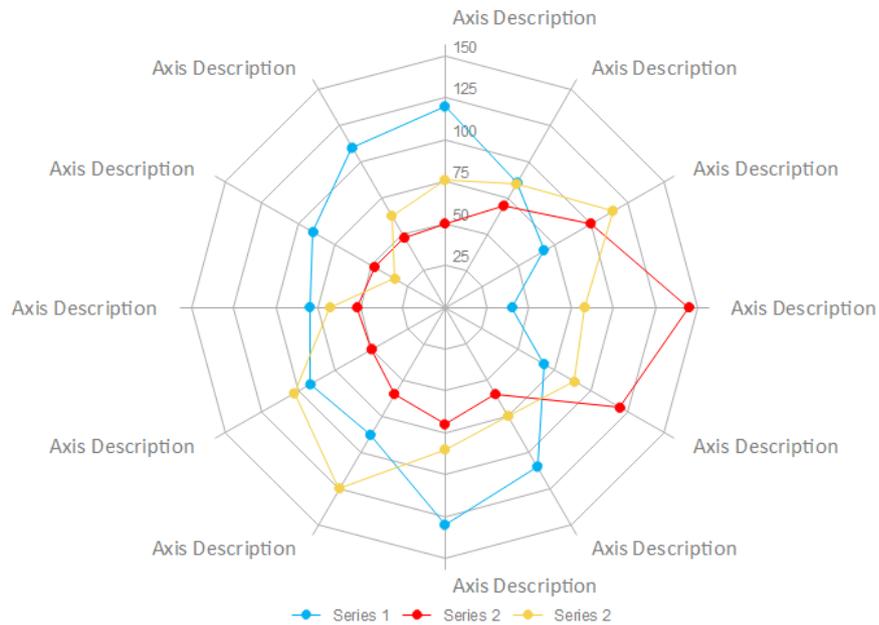


Figure 4.4: Radar Plot Schema [16]

Radar Chart

In the scatter chart of the K-means and in the scatter chart of the Hierarchical Clustering there are also red points (one for each cluster), which represent the centroid

of that cluster. These points are clickable and if the user clicks on it a new window shows him the radar plot of that centroid.

The radar plot represents the value of each component of the centroid (the arithmetic average of each attribute) on different Cartesian axes having the same origin [16]. These axes have different scales based on the minimum value and the maximum value of the attribute represented on it and the points on the different axes are joined with segments, so that the graph has the shape of a star or a spider web (see Figure 4.4). Each radar plot represents the centroids of all the clusters found, to allow the user to easily compare them; but only the segments that represent the centroid on which the user has clicked are highlighted in red (the others are blue).

Stacked bar chart for cluster validity

If a label is already present in the original dataset, this attribute is found with a semantic search for words like 'class', 'species', 'type', 'outcome'. A distribution of this attribute is shown to the user, who can graphically see how many values are present in each class.

Since a label is already present, it would make no sense to apply clustering algorithms (unsupervised learning) to try to divide our data into different classes. What we have decided to do, however, is to eliminate this class attribute and apply the various clustering algorithms anyway, then showing which of these algorithms give us a cluster division similar to the original class division of the dataset. The comparison between the original class division and the division obtained by the clustering algorithm is shown if the user clicks on the button 'Compare with the original label' (button present only if the search for the label has been successful). After this click, a new pop up window is opened, containing a stacked bar graph, that is a histogram that simultaneously shows both how the records are divided between the various clusters, and which of the original classes these records belong to.

Decision Tree

After assigning a label to each point thanks to a clustering algorithm, supervised algorithms can be applied to the dataset. In particular, by clicking on the button 'See

Decision Tree', the user can see, for each clustering algorithm, the decision tree built on his dataset.

In the decision tree shown to the user, inside each node there are some information regarding the split and the node itself (these information are reported in figure 4.5).

In particular, in each node there is:

- split: which tells us the split criterion
- gini: an index representing the degree of impurity of the current node. For low values the node is purer and therefore all the values tend to belong to a single class, while for higher Gini index values we have greater disorder and records tend to be distributed among different classes
- sample: represents the number of records that each node takes into account
- value: represents the number of records belonging to each of the different classes
- class: represents the class that best characterizes the node (the class in which the greatest number of elements is present)

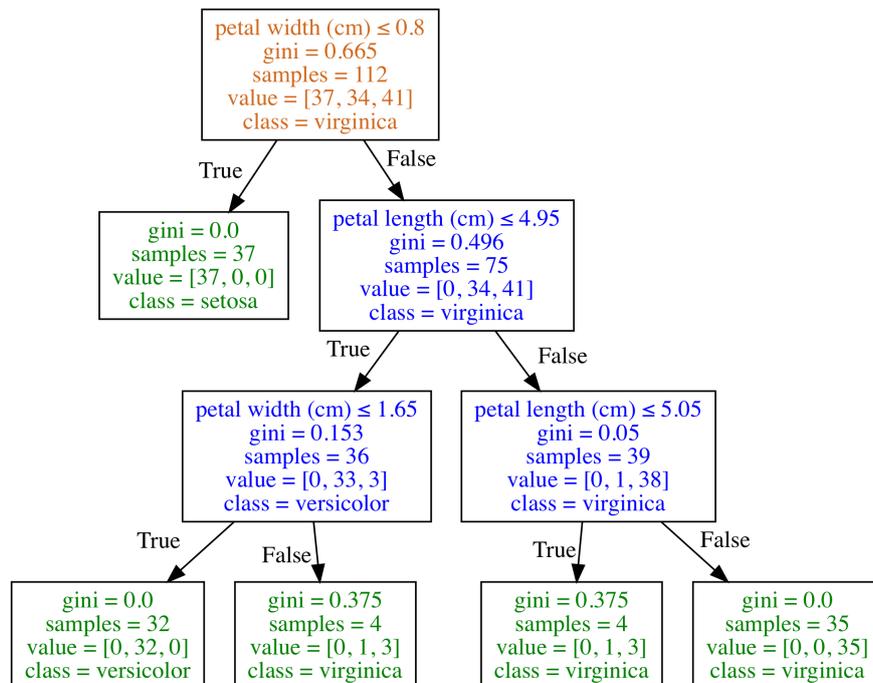


Figure 4.5: Example of a Decision Tree applied to the Iris dataset [14][17]

4.4 Data Transformation

In this section we wanted to show the user different possible transformations for the loaded dataset. By reporting the same data in a different structure than the original one, hidden information that was not very visible before, maybe can now become clearer. But such transformations are not suitable for all types of datasets. On some datasets it does not make sense to apply such techniques. And since our tool wants to be an automatic tool that works without the help of the user, to decide if it makes sense to apply these transformations on a given dataset or not, we searched among its attributes if there is one of a temporal type ('data', 'day', 'month', 'year'). If the search is successful, the user can click on this section and he can see the various proposed results in it. Otherwise the section 'Data Transformation' is not present in the vertical menu that is on the left on all the pages and consequently the user cannot access it.

Our idea to transform the dataset, changing its structure to emphasize new information, is to consider only two attributes among the initial ones. Let's suppose that the attribute 'A' has a number 'n' of distinct values and that the attribute 'B' has a number 'm' of distinct values, with $n < m$. Our new dataset will have m rows and n+1 columns. The first column will contain a row for each distinct value of B, the other n columns will each represent a value of A. The generic cell $x_{i,j}$ (i is the row, j is the column), will therefore contain a fraction obtained as the number of times the value of column j is associated with the value of row i in the original dataset, divided by how many times the value of column j appears in the original dataset.

If, just to give a practical example, we take a starting dataset like the one shown in the table 4.1:

id	eta	sesto	codpresta	codbranca	data
31	72	1	89.7	99	20070214
31	72	1	90.27	98	20070214
33	77	1	90.44.4	99	20070214
33	77	1	89.7	98	20070312
35	69	1	90.27	98	20070215
35	69	1	89.7	98	20070215
37	70	0	90.27	98	20070217
37	70	0	90.27	98	20070225

Table 4.1: *Initial Dataset*

and we want to transform it according to the Id-Codpresta attribute pair, the final dataset will be as the one shown in table 4.2:

id	Codpresta 89.7	Codpresta 90.27	Codpresta 90.44.4
31	0.33	0.25	0
33	0.33	0	1
35	0.33	0.25	0
37	0	0.5	0

Table 4.2: *Dataset After Transformation*

Now that we have understood how to transform our dataset, it remains to explain how to choose the two attributes to be used in the transformation. Generating all the possible transformations taking all the possible pairs of attributes doesn't make sense, because with large datasets the solutions shown to the user would be a very large number, but only a small part of them would be really interesting.

To decide which pairs of attributes to use for the transformation, we considered

each possible pair and we evaluated its behavior in the case the two attributes were merged into a single attribute. The operation of merging two attributes into one column is shown in the figure 4.6:

ID	Exam		ID_Exam
1	80	➔	1_80
2	80		2_80
2	81		2_81
3	81		3_81
3	81		3_81

Figure 4.6: The two attributes 'Id' and 'Exam' are merged into a single column

The idea is that, if the new attribute created behaves like an Id attribute, then the two starting attributes will contain interesting information and we can make the transformation of the dataset based on these features. After combining two attributes, merging them into a single column, we calculated its average frequency, as the total number of lines divided by the number of distinct values of the attribute. If this frequency is lower than a threshold we have set to 4.5, but it is greater than 1 (because we want our new attribute to look like an id, not to be a real id), then the transformation can be done. Before proceeding with the transformation it is also checked that the average frequency of the new attribute is less than the average frequencies of the two attributes that compose it (because otherwise the merging operation would have been useless). If we want to sum up what has been said up to now, we can use these two formulas:

$$1 < AverageFrequency_{New} < \Delta,$$

$$AverageFrequency_{New} < Min(AverageFrequency_{Attr1}, AverageFrequency_{Attr2})$$

where delta is a value found empirically by analyzing different datasets.

On the page relating to the 'Data Transformation' section, first of all the scheme of the original dataset is shown and the original distribution is displayed in a 3d scatter chart. Then, all the pairs of attributes that are considered suitable for processing are

listed. To see the transformation, the user must click on the name of the attributes. A new window will open; this window will contain the scheme of the new transformed dataset and a comparison between the 3d distribution of the original dataset and the 3d distribution of the dataset after the transformation.

The scheme of the transformed dataset contains only its first five lines, but, by clicking on a link next to the structure, the user can download the entire dataset in a csv format, thus having the new dataset available for further analysis.

4.5 StoryTelling

In this last section of the application, the most salient information regarding our uploaded data will be automatically show to the user. Structures and models hidden in the data are reported in a completely automatically way and the user must not interact anymore with the application. Furthermore, the information in this section is often also present in previous sections. But while in the other sections all the possible models and structures hidden in our data were shown in cascade, now we try to show only the most interesting solutions, that are those that mostly characterize the dataset. The purpose of this section is therefore to tell the story of the uploaded data, so that the user can have an immediate summary of it, even without analyzing all the other sections in detail. To provide this summary, the section is divided into several pages, each of which tells different analyzes to the user. Up to now the Storytelling section provides four different views named 'Data Characterization', 'Unsupervised Learning', 'Data Transformation', 'Supervised Learning'.

First view - Data Characterization

In the first of these pages there is a brief overview of our dataset, which includes the number of rows, the number of columns, the list of numerical and textual attributes. In addition to this basic information there is also a 3d scatter plot, which shows the distribution of the data, and the correlation matrix that contains the Pearson coefficients between the various numeric fields. This matrix is clickable, in fact clicking

on a cell, a pop up window is opened, which contains the boxplots and the Cumulative Distribution Function concerning the attributes related to the clicked coefficient. Obviously clicking on a cell of the diagonal (coefficient=1 between equal attributes), only one boxplot and one cdf is shown.

Second view - Unsupervised Learning

After the first page, the user can access a second one concerning clustering. In the clustering section 'Derived by the user' (cluster analysis described in 4.3), there were all the results of the various analyzes made with the parameters chosen automatically by the framework and later the user could see new results by manually changing these parameters. Now instead, in this second storytelling page, between all the solutions automatically shown by the framework and those manually selected by the user, only the best one is displayed (chosen using the Silhouette score).

This solution is represented as usual in a 3d scatter chart, where each cluster is identified by a different color. By clicking on a point belonging to a particular cluster, a new page is opened containing information regarding that cluster. On this page there is:

- a radar plot related to the centroid of the cluster (only if the best solution is k-means or hierarchical clustering)
- boxplots (one for each attribute) that show the distribution of the various attributes within that cluster
- decision rules, that is all the paths that can be extracted from the decision tree built on that solution. The paths that bring a point to be classified in the cluster clicked are highlighted in red

Third page - Data Transformation

There is also a third page that summarizes the results obtained in the 'Data Transformation' section. This page is obviously only present if a temporal attribute has been found. This page shows the three best transformations between those obtained, reporting for each of these the scheme of the new dataset and its distribution in a 3d

scatter chart. To choose which are the best transformations among those obtained, we apply to each of them a clustering algorithm (k-means with $k=3$). After doing this, we calculate the Silhouette score for each of these transformations and choose the ones with the highest score.

Fourth page - Supervised Learning

Finally, a fourth page is present only if the original dataset contains a label. In this case, the results of a series of classification algorithms applied to it are shown here. To apply these techniques, we first divided the data into two sets: the training set (which is used to train the model) and the test set, used to determine the accuracy of the model. But the risk of dividing the data in these two sets only once is to take (randomly) sets that are not really representative of the entire dataset and that therefore lead to finding inaccurate accuracy values.



Figure 4.7: Example of how k -fold cross-validation works [18]

To avoid this overfitting problem (lack of generalization of the model), we used k -fold cross-validation technique. With this technique the input data are divided into k subsets (in our case $k=10$). Nine of these subsets are used as training sets and the remaining subset is used as a test set. This operation is repeated k times, each time using a different subset as a test set. Each of these iterations will find a certain accuracy value of the model used; total accuracy is given by the average of the k accuracy values found. The classification techniques we have used are [19]:

- Support Vector Machine
- Decision Tree
- K-Nearest Neighbor (K=3)
- Naive Bayes Classifier

For each of these techniques there is a confusion matrix (see Figure 4.8), that is a matrix in which each row represents the predicted values, while each column represents the real values. The generic cell $x_{i,j}$ (i is the row, j is the column) represents the number of elements labeled i , which have been classified with a label j . If all the elements were classified correctly, the matrix would contain non-zero values only on the diagonal, while all the other values would be zero.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 4.8: Example of confusion matrix with two labels (positive and negative). True positive (TP) and True negative (TN) are the values classified correctly [20]

The sum of the values on the diagonal, divided by the sum of all the values of the matrix, constitutes the accuracy of the model. These accuracy values (one for

each algorithm) are shown alongside in a table, in which the row containing the algorithm with highest accuracy is highlighted.

Chapter 5

Results and evaluation

In this chapter we have decided to concretely describe how our application presents itself to the user. To do this, we will report below a series of images that are screens of our framework in the various phases of analysis of a dataset.

In particular, in order to report the results of all the analyzes that the framework makes available, we have separately analyzed two datasets. The first dataset used is called 'Iris', a free dataset available at the UCI Machine Learning Repository [17]. We have chosen this dataset because it is very simple and because, containing a label, it is possible to apply classification algorithms on it.

The second dataset of which we will report the results is called 'prescriptions' and contains a temporal attribute that allows us to view the content of the 'Data Transformation' section.

5.1 Iris Dataset

This dataset (available at <https://www.kaggle.com/uciml/iris>) contains 150 records, each of which represents a different flower. For each flower, numerical attributes are given, such as the length and the width of the sepals and petals, in centimeters. Each flower also belongs to one of these three species of iris (setosa, virginica, versicolor). In particular, there are 50 flowers for each species.

Table 5.1 shows the first four lines of this dataset:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa

Table 5.1: *Iris Dataset*

5.1.1 Data

First of all, after the upload page, the user is redirected to the page shown in the figure 5.1.

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4
7	4.6	3.4	1.4	0.3
8	5.0	3.4	1.5	0.2
9	4.4	2.9	1.4	0.2
10	4.9	3.1	1.5	0.1

Figure 5.1: Section 'Data'. First page shown. The user can scroll the table horizontally and vertically to see the whole dataset

5.1.2 User Exploration

In the 'User Exploration' macro section, the user can access two different sections: Info and Clustering. The following images show the information contained in the 'Info' section.

Info

In this section the user can see statistics and graphs related to the different types of attributes, both numerical and categorical, as shown in the figures 5.1, 5.2, 5.3, 5.4, 5.5, 5.6.

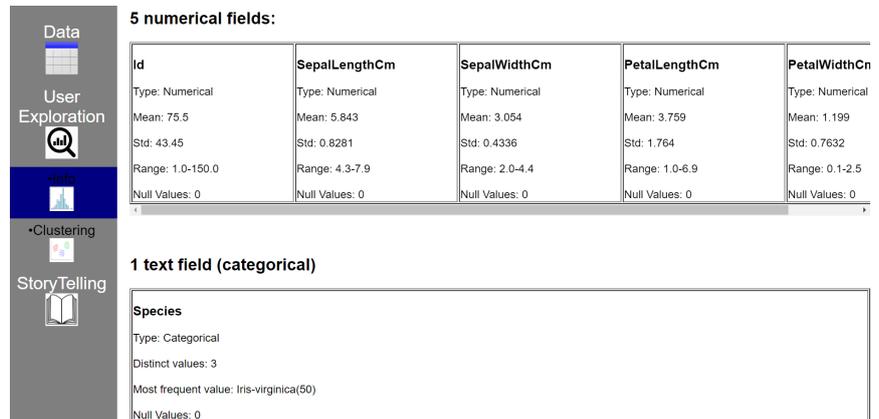


Figure 5.2: Section 'Info'. The user can see some statistics on the various attributes

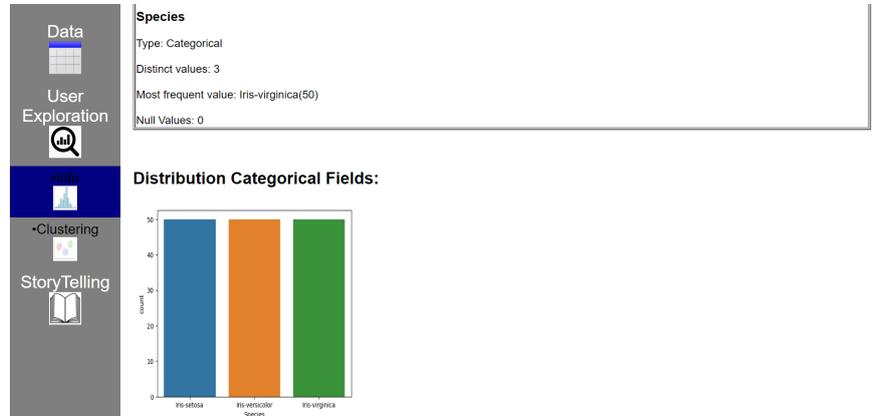


Figure 5.3: Section 'Info'. 'Species' is the only textual attribute. Histogram containing the number of elements for each species

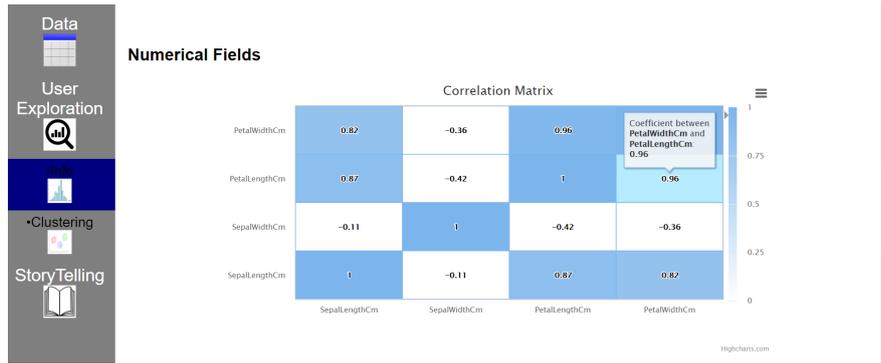


Figure 5.4: Section 'Info'. Correlation Matrix

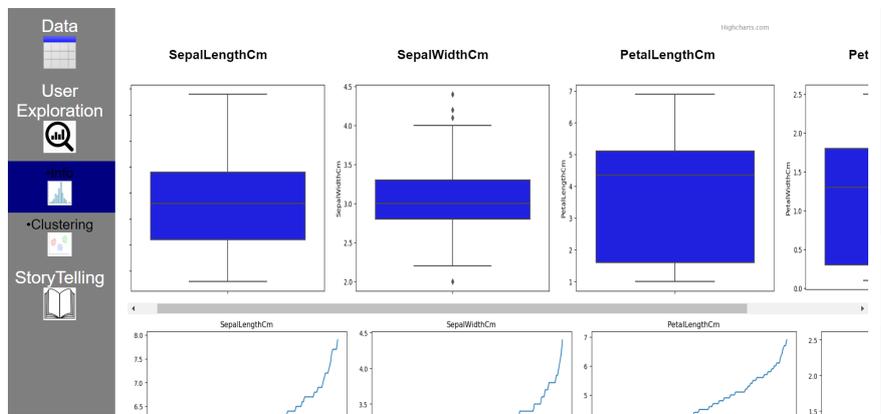


Figure 5.5: Section 'Info'. Boxplots for each numerical field

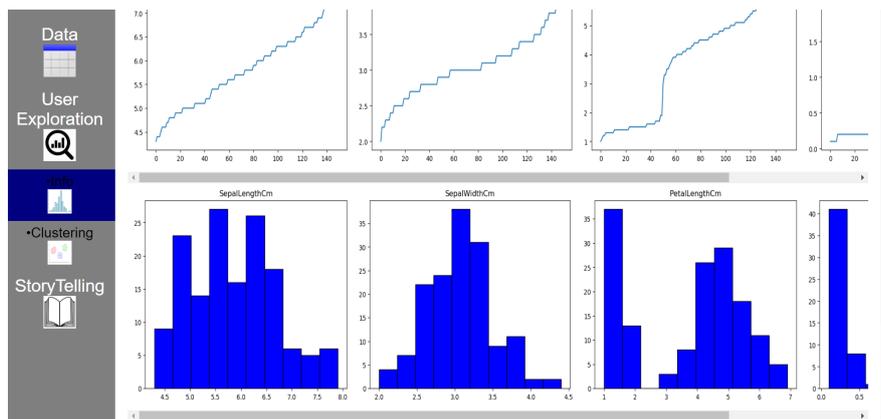


Figure 5.6: Section 'Info'. CDF and Histograms for each numerical field

Clustering

Here the user can graphically see the outliers present in the loaded dataset (Figure 5.7) and he can see the various results regarding cluster analysis, as shown in the

figures 5.8, 5.9, 5.10.

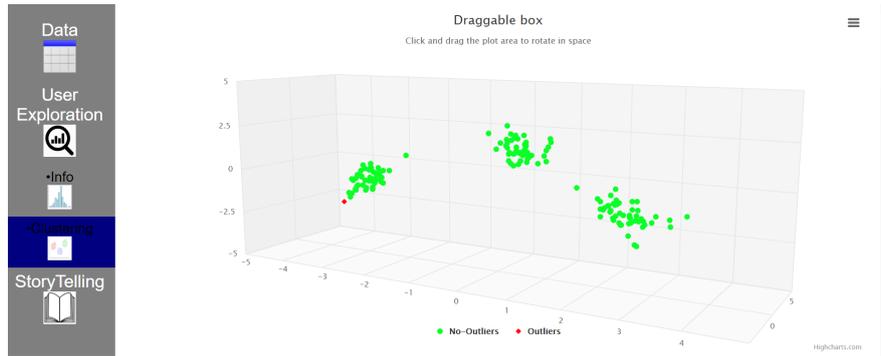


Figure 5.7: Section 'Clustering'. 3d scatter chart to show the outliers

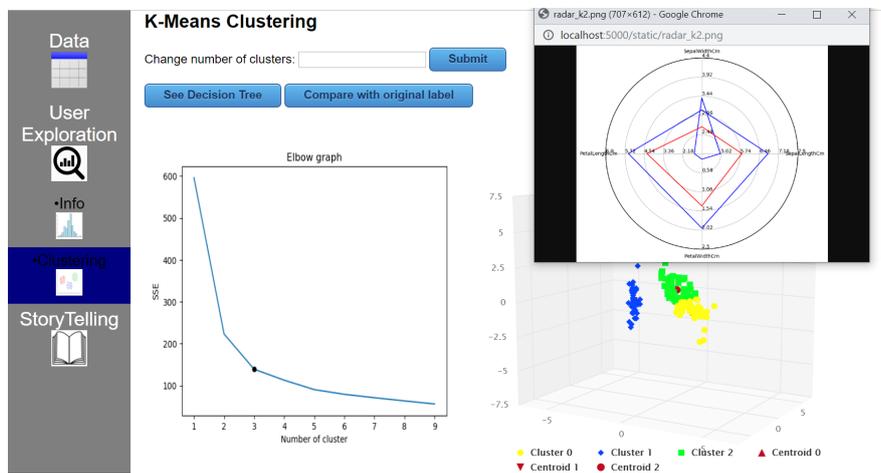


Figure 5.8: Section 'Clustering'. Elbow graph and 3d scatter chart for the K-means. The radar chart is obtained by clicking on the centroid2 on the scatter chart. To the right of the scatter chart there is also a pie chart

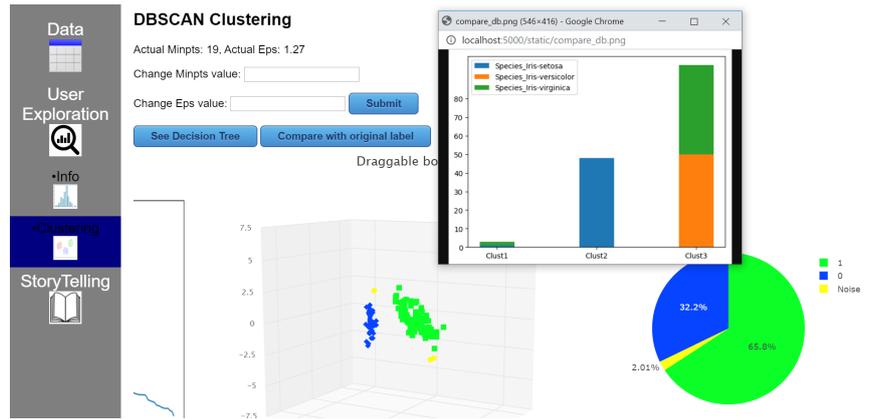


Figure 5.9: Section 'Clustering'. 3d scatter chart and pie chart for the DBSCAN. On the left of the scatter chart there is also a k-distance graph. The stacked bar chart is obtained clicking on the button 'Compare with original label'

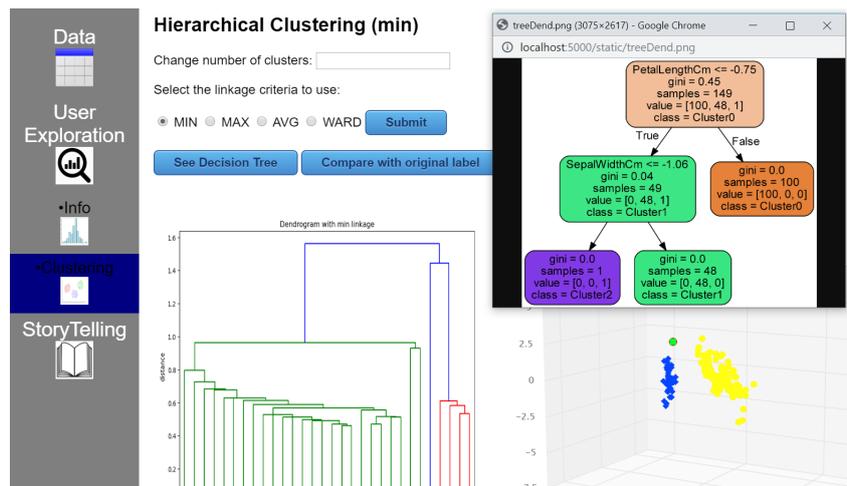


Figure 5.10: Section 'Clustering'. Dendrogram and 3d scatter chart for the Hierarchical Clustering. The popup window is obtained by clicking on the button 'See decision tree'

5.1.3 StoryTelling

This section contains three pages shown automatically: the Data Characterization page (Figures 5.11 and 5.12), the Unsupervised Learning page (Figures 5.13 and 5.14) and the Supervised Learning page (Figure 5.15).

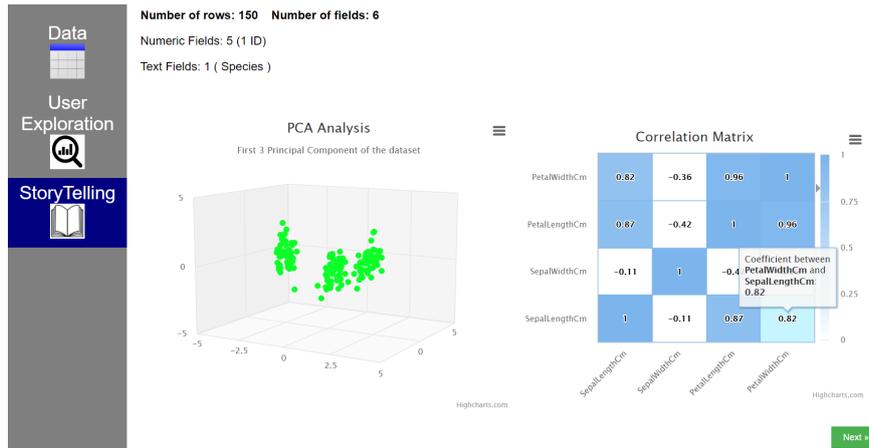


Figure 5.11: Section 'Storytelling'. First Page

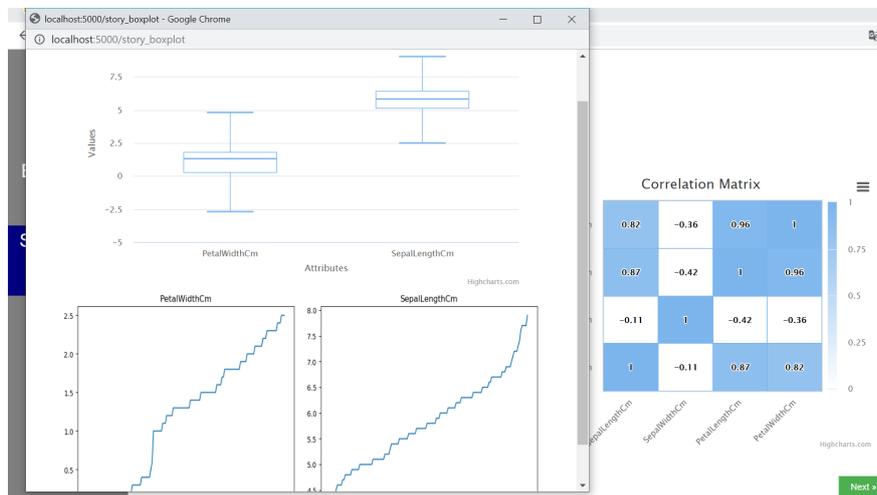


Figure 5.12: Section 'Storytelling'. First Page. Result obtained by clicking on the correlation matrix cell highlighted in the previous image

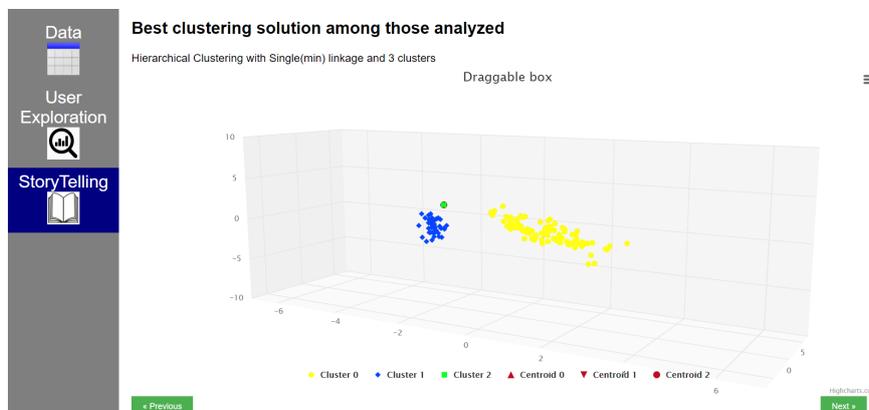
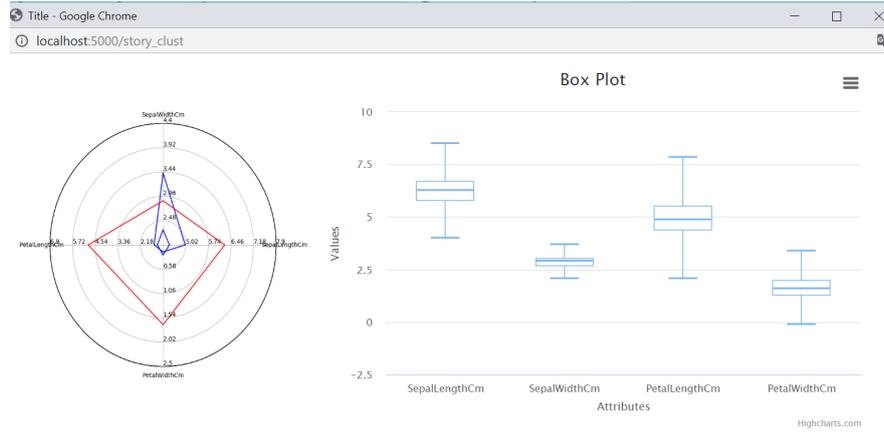


Figure 5.13: Section 'Storytelling'. Second Page



Decision Rules:
 if PetalLengthCm <= 2.45 AND if SepalWidthCm <= 2.6 ==> cluster 2
 if PetalLengthCm <= 2.45 AND if SepalWidthCm > 2.6 ==> cluster 1
 if PetalLengthCm > 2.45 ==> **cluster 0**

Figure 5.14: Section 'Storytelling'. Second Page. Result obtained by clicking on any point in the cluster0

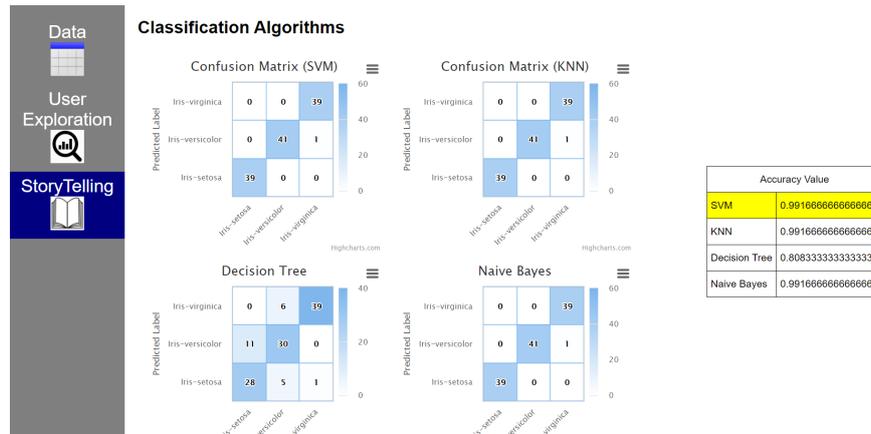


Figure 5.15: Section 'Storytelling'. Third Page

5.2 Prescription Dataset

This second dataset contains medical information. In particular, each line contains information about the exam (codpresta) made by a patient (ID), on a given date (data).

The first four lines of this dataset are shown in the table 5.2:

id	eta	sesso	codpresta	codbranca	data
31	72	1	89.7	99	20070214
31	72	1	90.27.1	98	20070214
31	72	1	90.44.4	98	20070214
31	72	1	91.49.1	98	20070214

Table 5.2: Prescription Dataset

5.2.1 Data

Figure 5.16 show the result of the 'Data' page, the first page shown after the upload.

'prescrizioni_rid_500' Dataset
Number of rows: 499 Number of fields: 6

id	eta	sesso	codpresta	codbranca
31	72	1	89.7	99
31	72	1	90.27.1	98
31	72	1	90.44.4	98
31	72	1	91.49.1	98
31	72	1	88.73.5	98
31	72	1	89.01	98
31	72	1	90.27.1	98
31	72	1	90.44.4	98
31	72	1	91.49.1	98
31	72	1	91.49.2	98

Numeric Fields
 Text Fields
 Rows with null values

Figure 5.16: Section 'Data'. First page after the upload page

5.2.2 User Exploration

In this case, since there is a temporal attribute (data), the macro section 'User Exploration' contains within it three sections, not just two as in the previous case (figure 5.17). Let's look at the differences compared to the analysis of the previous Iris dataset.

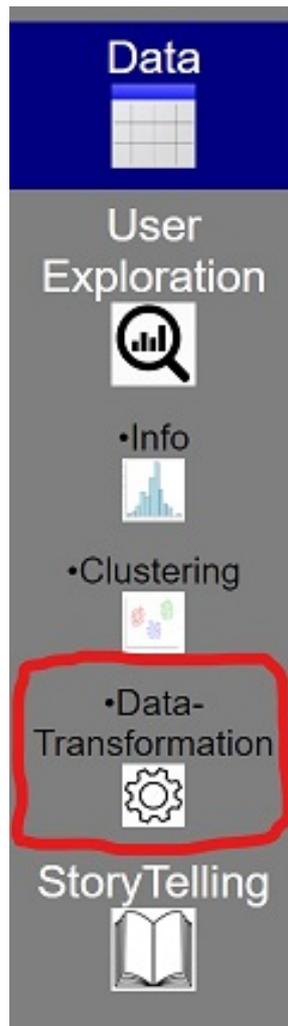


Figure 5.17: Vertical navigation bar present on all pages. In this case there is also the section 'Data Transformation'

Info

In this section, the information reported is the same as shown in the images 5.2, 5.3, 5.4, 5.5, 5.6 regarding the Iris dataset.

Clustering

Figure 5.18 shows the difference between the 'Clustering' section in the Prescription dataset and the analogous section depicted in figure 5.8 regarding the iris dataset.

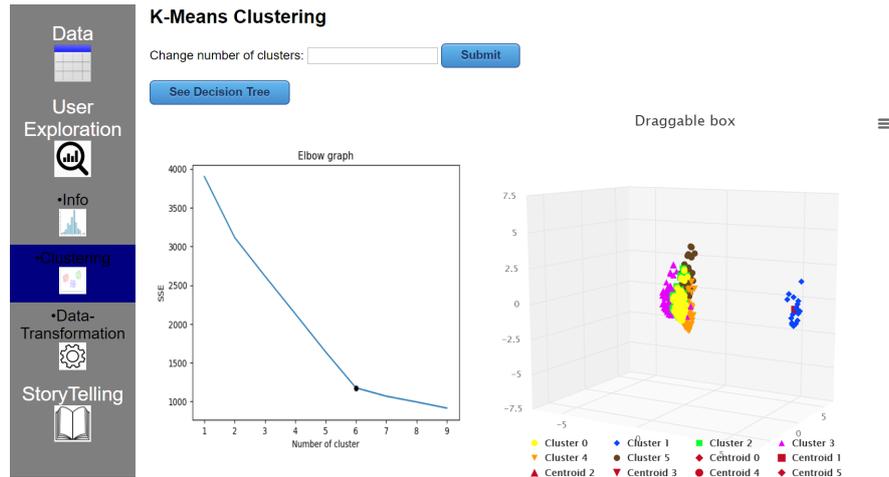


Figure 5.18: Section 'Clustering'. The structure of this section is the same as that reported for the Iris dataset. But in this case, since there is no label, the 'Compare with original label' button is no longer present

5.2.3 Data Transformation

Figures 5.19 and 5.20 depict how the possible transformations of the dataset are shown to the user.



Figure 5.19: Section 'Data Transformation'

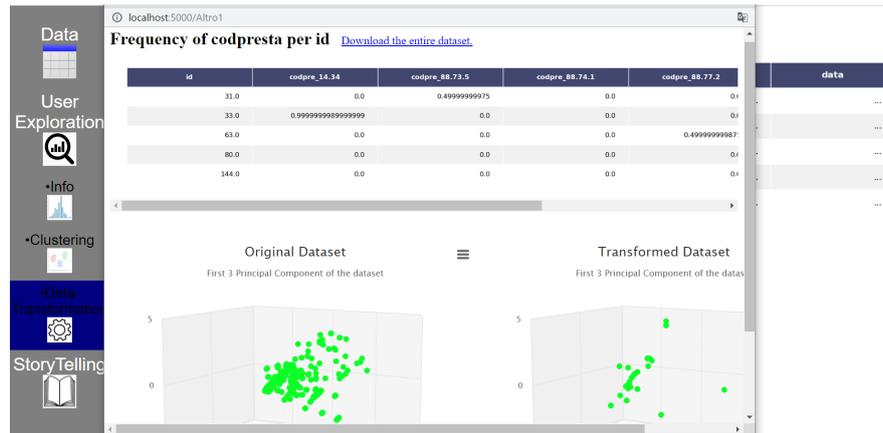


Figure 5.20: Section 'Data Transformation'. Result obtained after clicking on the transformation highlighted in the previous image

5.3 Storytelling

The different pages in the storytelling section are shown in the figures 5.21, 5.22, 5.23, and 5.24.



Figure 5.21: Section 'Storytelling'. First Page

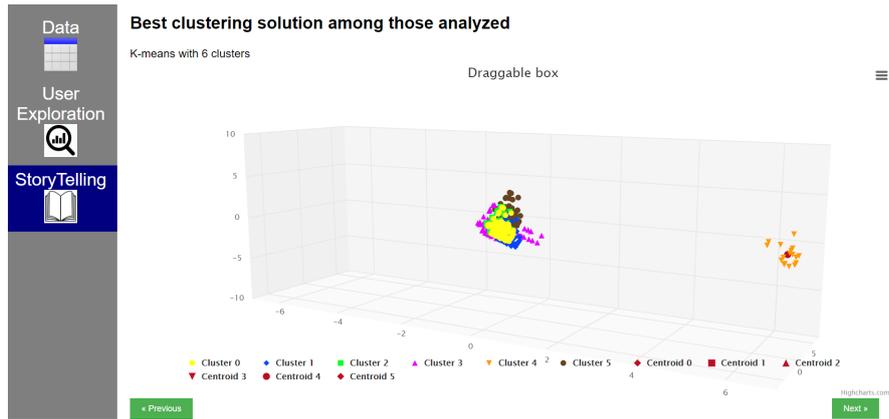


Figure 5.22: Section 'Storytelling'. Second Page

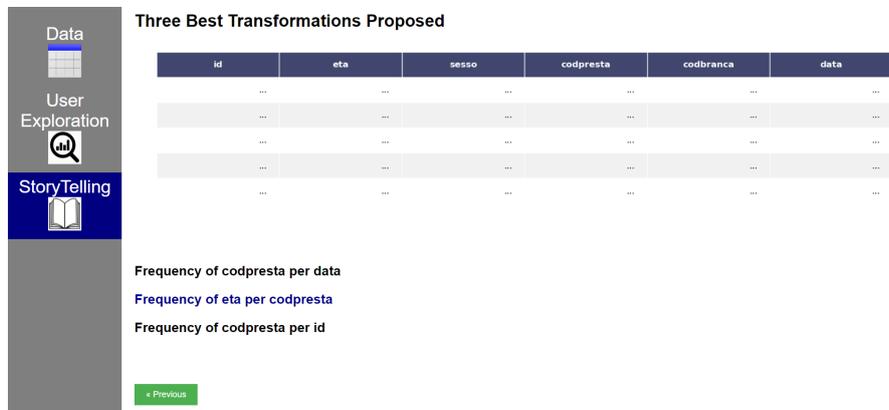


Figure 5.23: Section 'Storytelling'. Third Page

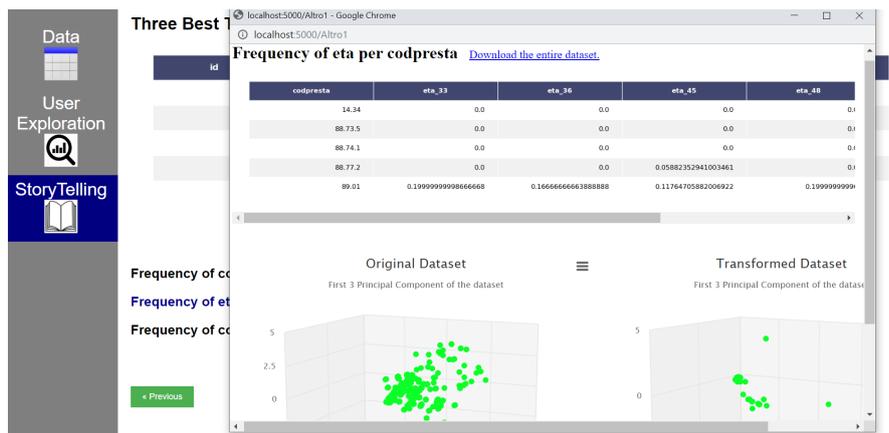


Figure 5.24: Section 'Storytelling'. Third Page. Window opened after clicking on the transformation highlighted in the previous image

Chapter 6

Conclusions

With this thesis work we have introduced our new framework for the automated data exploration. The work done up to now has allowed us to automatically analyze a large number of datasets, obtaining from them interesting graphs and statistics. All the results obtained by our framework are shown in the simplest way possible, so that all users, even the less experienced, can take advantage of our work.

Obviously what has been done in these months must be considered as a starting point. Automatic data exploration is a complex topic. And the idea of doing it automatically makes everything even more difficult. We need to find techniques and algorithms that can be generalized as much as possible, so that they can be applied without problems to any type of dataset. The results obtained so far are still very good, but we can make the work even more complete by adding different improvements. These possible improvements can be the following.

Extension of the supervised learning section

In the current framework the classification techniques are applied only in a page of the storytelling section. In the future we could create a special section of Supervised Learning, with a more detailed study on the choice of parameters for each of these techniques.

Time series analysis

This particular type of dataset requires a special preprocessing step that has not been implemented in the framework. But it may be added in a second moment.

Georeferenced Data

An additional task to add to the framework could be to manage the presence of georeferenced attributes, for example representing each element of this attribute above a map.

New Transformation Techniques

Only one type of transformation was proposed in this framework. And in which cases applying this transformation was decided based on empirical results obtained by analyzing different datasets. Maybe in the future we can deepen the study related to these transformation techniques, obtaining more accurate and more generalizable results.

Bibliography

- [1] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.
- [2] Joe Kuan. *Learning highcharts 4*. Packt Publishing Ltd, 2015.
- [3] Kevin Hu, Diana Orghian, and César Hidalgo. "DIVE: A Mixed-Initiative System Supporting Integrated Data Exploration Workflows". In: (2018).
- [4] Agustin Garcia Asuero, Ana Sayago, and Gustavo González. "The Correlation Coefficient: An Overview". In: *Critical Reviews in Analytical Chemistry - CRIT REV ANAL CHEM* 36 (Jan. 2006), pp. 41–59. DOI: 10.1080/10408340500526766.
- [5] Nadir Omer Fadl Elssied, Othman Ibrahim, and Ahmed Hamza Osman. "A Novel Feature Selection Based on One-Way ANOVA F-Test for E-Mail Spam Classification". In: 2014.
- [6] Michael Galarnyk. *Understanding Boxplots*. 2018. URL: <https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51> (visited on 09/12/2018).
- [7] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, et al. "Cluster analysis: basic concepts and algorithms". In: *Introduction to data mining* 8 (2006), pp. 487–568.
- [8] Kedar Potdar, Taher Pardawala, and Chinmay Pai. "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers". In: *International Journal of Computer Applications* 175 (Oct. 2017), pp. 7–9. DOI: 10.5120/ijca2017915495.
- [9] Hunter Heidenreich. *Natural Language Processing: Count Vectorization with scikit-learn*. 2018. URL: <https://towardsdatascience.com/natural-language->

- processing-count-vectorization-with-scikit-learn-e7804269bb5e (visited on 05/17/2019).
- [10] Chirag Deb and Siew Lee. "Determining key variables influencing energy consumption in office buildings through cluster analysis of pre- and post-retrofit building data". In: *Energy and Buildings* 159 (Nov. 2017). DOI: 10.1016/j.enbuild.2017.11.007.
- [11] Ville Satopaa et al. "Finding a "knee" in a haystack: Detecting knee points in system behavior". In: *2011 31st international conference on distributed computing systems workshops*. IEEE. 2011, pp. 166–171.
- [12] Manoj Singh. *Finding the elbow or knee of a curve*. 2017. URL: https://dataplatform.cloud.ibm.com/analytics/notebooks/54d79c2a-f155-40ec-93ec-ed05b58afa39/view?access_token=6d8ec910cf2a1b3901c721fcb94638563cd646fe14400fecbb76cea6aaae (visited on 10/19/2017).
- [13] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.
- [14] Michael Galarnyk. *Understanding Decision Trees for Classification (Python)*. 2019. URL: <https://towardsdatascience.com/understanding-decision-trees-for-classification-python-9663d683c952> (visited on 07/31/2019).
- [15] Fred B Bryant and Paul R Yarnold. "Principal-components analysis and exploratory and confirmatory factor analysis." In: (1995).
- [16] Edraw. *When to Use a Spider Chart*. 2019. URL: <https://www.edrawsoft.com/chart/when-to-use-spider-chart.php> (visited on 05/12/2019).
- [17] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: <http://archive.ics.uci.edu/ml>.
- [18] Sebastian Raschka. "Model evaluation, model selection, and algorithm selection in machine learning". In: *arXiv preprint arXiv:1811.12808* (2018).
- [19] Charu C Aggarwal. *Data classification: algorithms and applications*. CRC press, 2014.

- [20] Allison Ragan. *Taking the Confusion Out of Confusion Matrices*. 2018. URL: <https://towardsdatascience.com/taking-the-confusion-out-of-confusion-matrices-c1ce054b3d3e> (visited on 10/10/2018).

