ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA

SCUOLA DI ECONOMIA, MANAGEMENT E STATISTICA
Corso di Laurea in Scienze Statistiche

UNSUPERVISED ANALYSIS OF DIDACTICAL DATA

Presentata da:
Gaetano Romano
matricola: 0000727724

Relatore:
Prof Claudio Sartori

APPELLO 10-11-12/07/2017
ANNO ACCADEMICO 2016/17

Summary

Abstract

The following paper describes an unsupervised analysis of didactical data from University of Bologna. Specifically, data regards academic careers of students from MSc in Informatic Engineering. Since there are only few mandatory courses, the choice of which courses follow is left to the student. The aim of the analysis is to find, given students careers, if there exist any tendencies or specific educational plans.

Employed techniques are described in chapter 2, results in chapter 3. Last chapter, finally, describe the R implementation of the analysis, in case of future implementations.

The entire code and additional graphs can be found in the following repository[1].

[1] https://github.com/alghul96/ThesisAnalysis

Unsupervised Analysis of Didactical Data

1. Data Preparation

Data were received by the end of April. They were organized in two different datasets, one containing the careers of already graduated students from academic year 2012/13 to academic year 2016/17, the other containing careers of ongoing students.
The 6026 instances of the initial dataset represented one exam completed by one student.
The id of the student, the id of the exam and the exam name were some of the key information associated to each instance.

1.1 Creating a dataset for the analysis

The first part of the work consisted in creating a new analysis-ready dataset. Since cluster analysis had a key role in the project, the original data were organized in a sparse matrix.
In the matrix, the rows represented the students, while the columns represented the exams.
In correspondence of student i and exam k, the matrix contained the following:

$$v_{i,k} = \begin{cases} 1, & if\ student\ i\ passed\ the\ j\ exam \\ 0, & elsewhere. \end{cases}$$

The initial matrix consisted in 322 students, and 200 exams. From that, explorative analysis showed different problems in the way exam names were registered. After data manipulation, the final matrix with 322 rows and 102 columns was obtained and converted into a data frame. While the number of students remained unchanged, 96 columns were removed from the analysis. Moreover, additional data, such as the enrolment year, were added to a secondary data frame.

1.2 Data manipulation

To obtain the data frame ready for the analysis, various changes were made. The following changes were identified exploiting the domain knowledge from the data owner:

- Each course ending by "LS" (*i.e., Laurea Specialistica*) was replaced with the correspondent course ending for "M" (*i.e., Magistrale*);
- All the courses from the Bachelor degree were removed: they were from students that had to integrate different base courses from the bachelor degree in Informatic Engineering to enrol for the MSc degree;
- Some courses that changed the name or the id during years, but regarded the same topics, were merged together;
- All the language courses were removed;
- All the exams from other universities (*e.g.,* courses from Erasmus exchanges) in which it was impossible to determine the programme were merged into one single exam;
- Outliers and errors were cleaned.

2. Clustering Methods

Different clustering methods were considered in the analysis. From a large variety of methods, the following lead to better results.

2.1 Agglomerative Clustering Methods

Agglomerative hierarchical clustering methods aim to build a hierarchical structure in which are contained all the n instances (Kaufman, 1990). At the end of the clustering procedure this structure can be represented graphically via a tree called *dendrogram*. (Ian H. Witten, 2011). No assumption about the best number of cluster nor the distribution of the variables are made via these methods.
The first step is to obtain a dissimilarity matrix, containing dissimilarities for each instance in the dataset. Now, in our case we deal with asymmetric binary variables, and the dissimilarity was computed via *Jaccard coefficient.* Those can be computed via `daisy` function (Kaufman, 1990). This means that if two students, i and j, had the same *positive* value ($v_{i,k} = v_{j,k} = 1$) on the k exam, the similarity should count more on the dissimilarity coefficient than if they both hadn't passed the k exam. So, the dissimilarity matrix element $d_{i,j}$ is:

2.1

$$d_{i,j} = \frac{b}{a+b}$$

Where:
a is the sum of all the times two students have passed the same exam, $v_{i,k} = v_{j,k} = 1$;
b is the sum of all the time two students do not have the same value on an exam.

Considered a dissimilarity matrix, next step is to define dissimilarity between clusters. In a stepwise procedure, merging the clusters is evaluated based on their dissimilarity. At the first step, each student is considered as a separate cluster. Step after step, the clusters are merged together according to the nearest one in the dissimilarity matrix. The only difference between the agglomerative clustering methods is in the way we consider the distance between clusters. In the analysis the complete method, and the ward method were those that gave best results.

2.1.1 Complete linkage

The simplest method with best results that was used was the Complete linkage method. This method defines the distance between two clusters as the dissimilarity between two farthest points in a cluster. Let *R* and *Q* be two clusters. Then, the distance between *R* and *Q* is defined as:

2.1

$$d_{R,Q} = \max d_{i,j}\,, \qquad i \in R, j \in Q$$

Single linkage usually leads to many spherical clusters, often with a small within cluster dissimilarity. For this reason, this cluster method is said to be *space dilating* since relatively similar objects usually stays in different clusters (Kaufman, 1990).

2.1.2 Ward method

The best results were from Ward method. Ward method try to minimize the variance within clusters and merging groups that have the largest variance between.
Ward method computation, though, is based on Euclidean distances, and since our dataset contained only Binary variables the computation of Euclidean distance could lead to misleading results. For this reason, `agnes` algorithm was used. The algorithm is described into Kaufmann 1990, and it can start from a dissimilarity matrix, as the one we already have computed[2]. From that, the squared distance between cluster *R* and *Q* is computed as:

2.2

$$d_{R,Q}{}^2 = \frac{|A| + |Q|}{|R| + |Q|} d_{A,Q}{}^2 + \frac{|B| + |Q|}{|R| + |Q|} d_{B,Q}{}^2 - \frac{|Q|}{|R| + |Q|} d_{A,B}{}^2$$

Where:

- Cluster *A* and *B* are the two cluster that forms cluster *R* at the next level;
- Distances at the base level of the process are taken from the dissimilarity matrix, hence at the first step each unit is considered as a cluster.

It is important to remember that the distance 2.2 is obtained from the following *Euclidean-based* distance:

2.3

$$d_{M,N}{}^2 = \frac{2|M||N|}{|M| + |N|} \|\bar{x}_M - \bar{x}_N\|^2$$

Luckily, once computed the dissimilarity matrix via *Jaccard Coefficient* for our instances, for the distance formula 2.2 computation we will only use the already computed distances, so no Euclidean distance should be included in the process.

[2] See the beginning of paragraph 2.1

2.2 Partitional Clustering Methods

Partitional clustering methods start from a desired number of *k* clusters, the algorithms then initiate *k* centres and finally assign each instance to one of the specified clusters.
In the analysis, K-means and K-medoids algorithm were considered: both produce similar results since they're strictly related and differ only in the way cluster centres are specified (Wikipedia, 2017).

2.2.1 K-Means Algorithm

K-means is an iterative distance-based algorithm. It starts from choosing at random, in the dataset, *k* instances as the *k* centres of the clusters. Then, simply as that, it assigns the remaining $n - k$ units to the closest centroid, according to Euclidean distance.
The problem is that the first clusters are sensitive to the initial choice of the centres (Ian H. Witten, 2011). To avoid that, after the first iteration, new cluster centres are initiated. This is simply obtained via computation of the centroids of the instances in each cluster. Then, the process is repeated over and over until the centres of the clusters have converged to a final value.
Now it is proven that, after the iteration stabilize, the total squared distance between each cluster and all its point is at a *local minimum*.
To increase the chances of finding a *global minimum*, the whole algorithm can be repeated several times with different initial choices, and at the end keep the clustering that scores the smallest total square distance.

It can be stated that K-means should not be applied to our dataset, since results are obtained via computations of the clusters average[3] and via the research for the smallest total square distances. For this reason, K-means algorithm assumes numerical data.
Even if, treating our binary variables as numeric, the algorithm produced reasonable results, the K-medoids algorithm was applied.

2.2.2 K-Medoids algorithm

K-Medoids works similarly to K-means: they both are iterative partitional algorithms. The algorithm proceeds in the same way as K-means do, but after the first iteration, instead of computing new cluster centroids, it uses *medoids* as new cluster centres.
A medoid is defined as the object whose average dissimilarity to all objects in the cluster is minimum (Wikipedia, 2017). In practice, the medoids is the closest located instance to the cluster centroid.
Moreover, the algorithm minimizes a sum of pairwise dissimilarities instead of the total square distance.
Again, in our case, the two algorithms generated similar clusters. In K-medoids, cluster centres can be interpreted much easier than K-means centres.

[3] The average is used to obtain the second iteration centroids.

2.3 Probability-Based Clustering Methods

Until now we have only considered methods that assign *categorically* one instance to a given cluster, without any measure of uncertainty. Probability-Based clustering methods, on the other hand, look for *the most likely* set of clusters given our dataset. Hence, for every instance *i* we'll have *k* probabilities of belonging to each cluster.
Unfortunately, Probability-Based Clustering, as every method that includes uncertainty, is not distribution-free, and it follows a Bayesian approach, since each cluster is modelled by a probability distribution (Stutz & Cheeseman, 1996). Attention is needed to model those distributions, since they need to reflect the behaviour of our data.

2.3.1 Bayesian Bernoulli Mixture Model

For probability-based clustering we use the finite mixture statistical model. A finite mixture is a set of *k* probabilities distributions, one for each cluster. Each distribution is said to be a component of the mixture model. Each mixture component tells the probability that a certain instance *x* will have a set of attribute variables if belonging to that cluster. (Ian H. Witten, 2011)
Bayesian Mixture is needed for modelling our binary space. So, simplifying, the model that generates a vector of instances $\boldsymbol{x}$ is modelled as (Maaten):

$$f(\boldsymbol{\theta}) = Diriclet(\boldsymbol{\theta})$$

$$p(\boldsymbol{x}|\{\boldsymbol{\theta_1}, \dots, \boldsymbol{\theta_K}\}) = \sum_{k=1}^{K} (Bern(\boldsymbol{x}|\boldsymbol{\theta_k}))^{p_k}$$

The scope of this procedure is, given our set of n instances, and a specified number of cluster, *i.e.* mixture components, to specify the parameters that regulates our mixture model and the probabilities p_k of each instance to come from each mixture component.
Moreover, to penalize the model for introducing many components (one for each cluster), a prior probability for each distribution is introduced.
Since we're working with binary data, we will have $2k$ set of parameters to estimate, and a set of *nk* probabilities to find.
But we do not know any of this information. Hence, an iterative procedure is required.

2.3.2 Expectation Maximization procedure

The algorithm starts giving an initial guess for the unknown parameters $\boldsymbol{\theta}$ of our mixture model. Then, it calculates the cluster probabilities for each of our instance. This is told to be the *expectation* step, since we look for the expected cluster values. The second step is called the *maximization* step, because we estimate again the parameters of the distributions via the maximum likelihood estimator for our parameters $\boldsymbol{\theta}$.
Then, iteration continues until a pre-specified score based on computation don't change significantly from one iteration to the next. In the implementation of this algorithm used in our analysis a *lower bound* is computed as score: it is based on both on parameter values and classes

probabilities.
The EM algorithm converges to a maximum, but as in K-means procedure, it is not guaranteed that it reaches a global maximum. To overcome this, procedure is repeated more than one time. One of the advantages of this model is that it will automatically look for the best classification of our units, between different number of clusters, given a reasonable computation time.

2.4 Implementation of the Mentioned Methods

To implement the mentioned methods in the analysis, the following software and packages were used. For details and to implement analysis again, see Chapter 4.

- For Agglomerative Clustering Methods, were used both DAISY and AGNES algorithms, programmed originally in Fortran for the IBM, and implemented in R[4] via cluster [5] package, the same package was used for K-medoid algorithm;
- For K-means algorithm the R package flexclust [6] was used;
- For Bayesian Bernoulli Mixture Model the Python [7] package sklearn-bayes[8] was used for computation and modelling. It was implemented in R via PythonInR [9]package.

[4] R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

[5] Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., Hornik, K.(2016). cluster: Cluster Analysis Basics and Extensions. R package version 2.0.5.

[6] Friedrich Leisch. A Toolbox for K-Centroids Cluster Analysis. Computational Statistics and Data Analysis, 51 (2), 526-544, 2006.

[7] Copyright (c) 2001-2016 Python Software Foundation.
All Rights Reserved.

[8] https://github.com/AmazaspShumik/sklearn-bayes

[9] Florian Schwendinger (2015). PythonInR: Use Python from Within R. R package version 0.1-3. https://CRAN.R-project.org/package=PythonInR

3. Analysis

The following chapter will regard different phases of the analysis of our dataset and the results obtained. We will begin with a brief comparison of the different clustering procedures. After that, results will be discussed.

3.1 Comparison of clustering procedures

The most efficient clustering methods for our analysis were descripted in the previous chapter. Results obtained with those techniques were relatively similar. To show that, we take the best classification obtained, the one with 3 clusters. The reasons behind 3 clusters and the composition of those clusters will be discussed later in this chapter.
In Figure 3-1 we can see the dimension of each cluster for different clustering procedures. We can see that every procedure reaches similar results even in the number of student belonging to each cluster.
Looking at the dendrogram in Figure 3-3 we can have an idea on the repartition of our instances in our dataset. We see the 3 distinct groups, in which the smallest one, on the left, is the most different from the other two.
Figure 3-2 shows a comparison between the 3 centroids of K-means, the 3 cluster prototypes from Mixture Modelling clustering and lastly the 3 medoids of K-medoids. In yellow we found the exams that a typical student of each cluster is expected to follow. We can see that the composition of those clusters is similar between all the procedures.

3.2 Cluster description and composition

Results of the obtained cluster classification will be discussed, from the reason behind the choice of 3 clusters to the composition of each cluster itself.

Figure 3-1

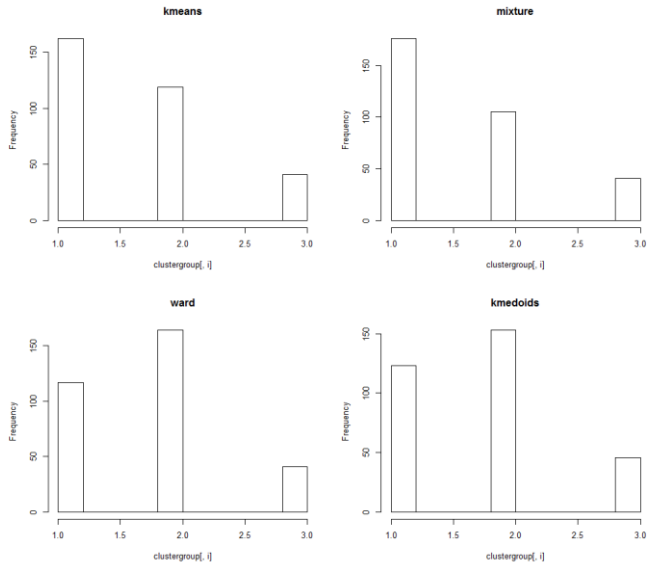


Figure 3-2

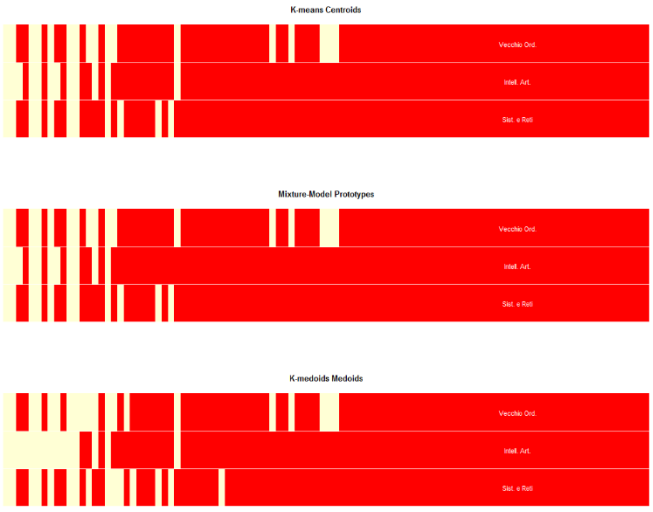
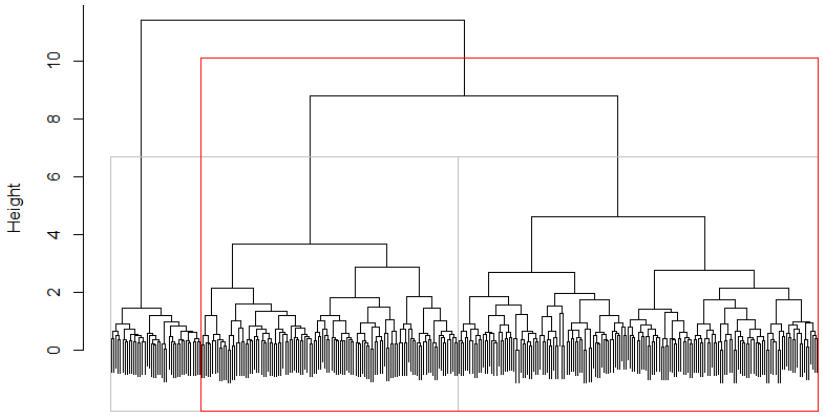


Figure 3-3



3.2.1 Choosing the number of clusters and quality of our clustering

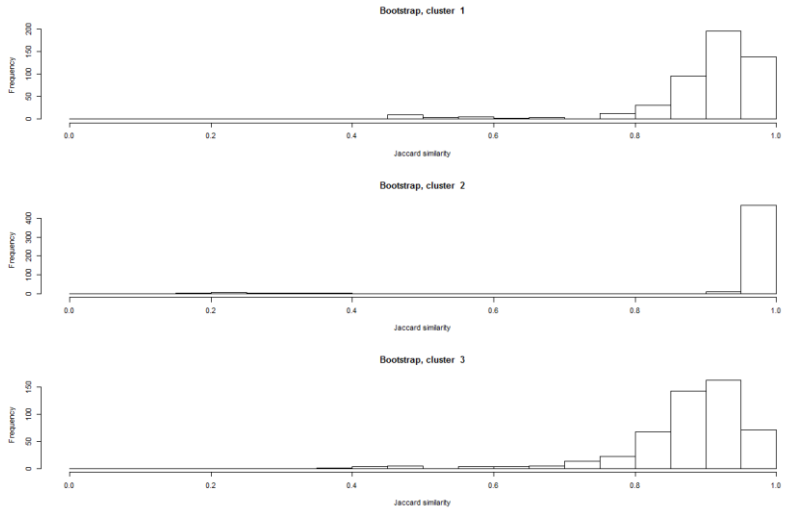
The final choice of 3 clusters was made differently for each procedure.
Bayesian Bernoulli Mixture Model clustering can look for best classification of our instances considering different number of clusters. On a basis of 5000 initialization of the algorithm, the best choice for our classification lead to 3 clusters.
For k-means and k-medoids, sum of within cluster distances and average silhouette width were used. Comparing different dimensions, the best choice was of 3 clusters again. However, the average silhouette width was, in both cases, remarkably low: this is because our classification might be affected by what is commonly known as Course of dimensionality (Wikipedia, 2017).

For this reason, a different cluster evaluation technique known as Bootstrapping (Hennig, Cluster-wise assessment of cluster stability, 2007) was used. To sum up, the procedure compares via Jaccard similarity coefficient the clusters obtained via clustering in the original dataset and the cluster obtained with the same cluster strategy, but on a resampling of the original dataset. This is because, if there is a tangible structure in data, even after a resampling, the mentioned structure should be maintained. Repeating the procedure several times, the average of the Jaccard similarities coefficient can be used as an indicator of cluster stability. The result for our clustering can be found in Figure 3-4.
A highly stable cluster is considered to have an average of Jaccard similarities coefficient above 0.8 (Hennig, 2008), which is our case for all the 3 clusters.

3.2.2 Composition of the 3 clusters

To get an idea of what the majority of students followed inside each cluster, the frequencies of each exam for each single cluster were computed. From those frequencies was possible not

Figure 3-4



only to understand which exams the student of each cluster followed, but also to define differences between the 3 clusters. The 3 clusters were identified as follows:

- *Old educational plan:* Those students form the smallest cluster; they are those who enrolled before 2012, the year in which all the plans were reformed. They differ from the others in the number of mandatory exams, that was drastically reduced in the reformation.
- *Artificial Intelligence:* Those students follow exams of topics concerning artificial intelligence, data analysis, computer vision and image processing and the related projects.
- *Systems and Networks:* As the name may suggest, those students follow courses regarding different system structures, such as distributed computing, mobile systems, computer networks and the related projects.

It is important to notice that the last two groups, even if different, have some exams that, even if not mandatory, were highly followed for both, such as: *Fundaments of Artificial Intelligence*, *Information's Security, Real-Time computing systems*.

3.3 Analysis on a reduced Dataset

From now on, the analysis focuses on a reduced dataset, where the students from the first group were removed alongside with their exams. By removing the first group it is much easier to focus on the major two groups. Moreover, also mandatory exams were removed, to improve visualizations.

3.3.1 Comparison with previous results

From the previous dataset, results were not so different. Logically, all clustering procedures brought to 2 clusters, *Artificial Intelligence* and *System and Networks*, the ones that we have mentioned before. As already said, with less exams and students, visualizations improved. In

Figure 3-5

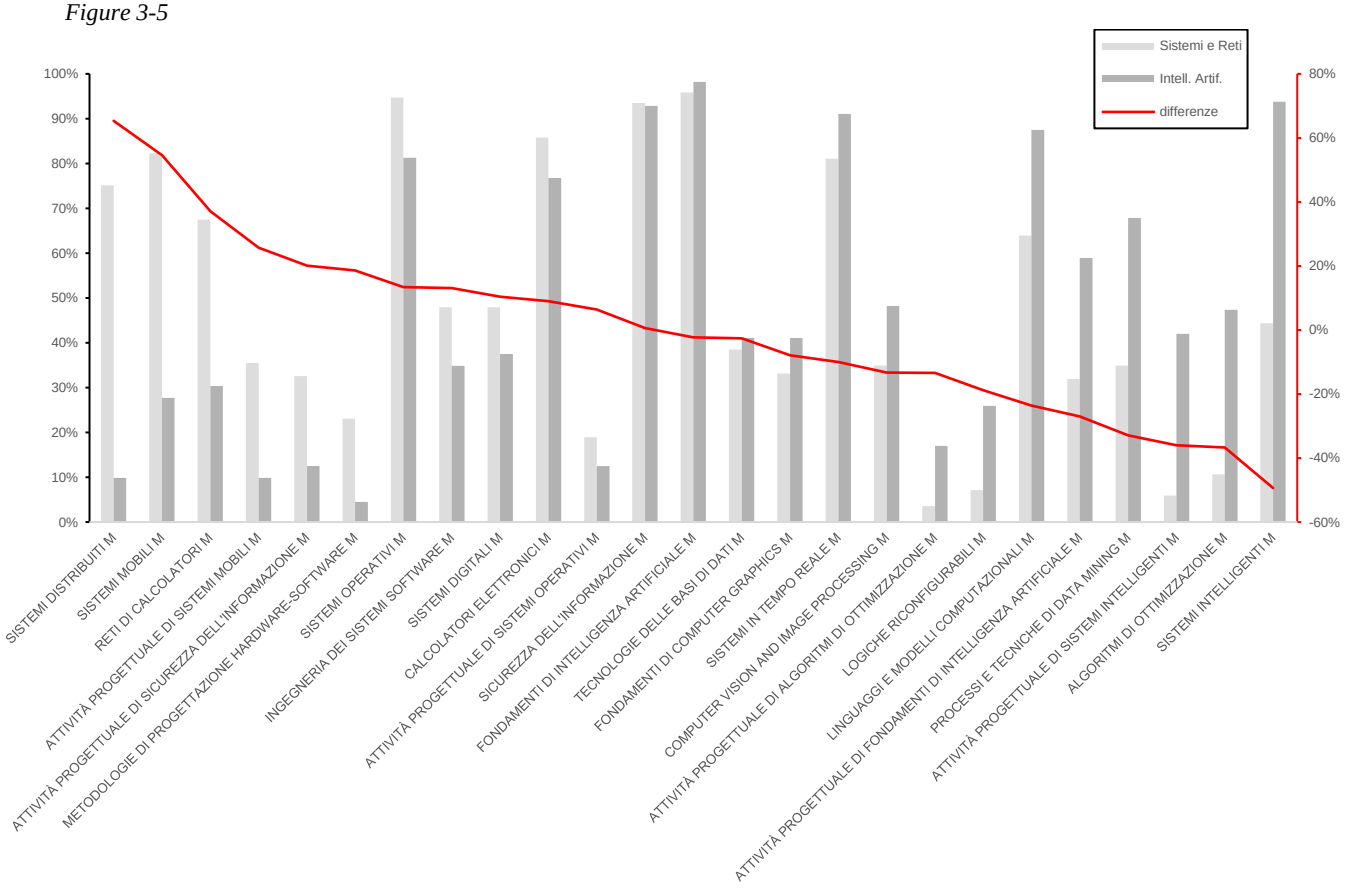


Figure 3-5 shows a histogram of the most followed courses from the 2 main clusters. We see from the left to the right, with the Italian denomination, respectively courses from System *and Networks* and *Artificial Intelligence;* in the middle we find exams which are followed in both. The line shows how much the two clusters differs on various exams.

3.3.2 Analysis for more than two clusters

For the analysis, number of clusters found on the reduced dataset was forced to be more than two. This was obtained via specifying different initial centroids for k-means and k-medoids clustering methods. As expected, the results, if evaluated with Bootstrapping procedure, lead to unstable clusters, with the average of Jaccard similarity coefficients of below 0.7.
Interesting results were found for the following 4 clusters:

- *Systems and Networks:* Similar cluster to the parent one described in the previous section.
- *Systems and Artificial Intelligence:* Students from this cluster tent to be hybrids from the 2 parent clusters. They both have exams in data science and distribute computing.
- *AI and Data Science:* Cluster containing students following exams regarding Artificial Intelligence and data science, database management and optimization.
- *AI and Computer vision:* Like the previous one, but more focused on computer vision and image processing rather than data science.

4 Description of the R Script for future analysis

Until now data regarding students from Informatic Engineering were analysed. The following chapter aims to describe the R implementation of the analysis, to make easier to repeat what already seen also on different dataset or for future analysis.

4.1 Script organization

These scripts are needed to perform the analysis:

- *auto_analysis.R:* script where a simplified version of the whole analysis is performed. It calls function from the following scripts;
- *external_functions.R*: script containing all the functions needed for different tasks, from data manipulation to performing the clustering;
- *simpleModelbased.py, mixture_models.py:* Python 2.x scripts needed only to perform the probability based clustering. In order to run the following, Python 2.x is required, even though also Python 3.x might work, however, it has not been tested.

4.2 Preparing the dataset for the analysis

A binary dataset is required to perform the analysis. To obtain the following dataset, function `convert_dataframe()` should be used. It takes as argument a dataset in the form of the one in Table 4-1, and it converts it in an analysis-ready dataset.

Table 4-1

Matricola	Esame (codice)	Esame (Denominazione)
0000648539	35223	CALCOLATORI ELETTRONICI M
0000648539	29206	GESTIONE DELL'INNOVAZIONE E…
0000648539	35262	SISTEMI MOBILI M
…	…	…

To check if everything is ok, an image of the dataset should look like Figure 4-1.

4.3 Performing the analysis

4.3.1 Clustering methods

To simplify everything, function `studyplan_finder(…)` performs the analysis for all the previous mentioned clustering methods[10]. It is specifically designed to work on a sparse binary dataset.
A complete list of the arguments it takes can be found in Table 4-2.
The output of the function is a collection of objects of different classes, based on the selected methods for the analysis. E.g.: to reach k-means output, `studyplan_finder(…)$kmeans`.

[10] See Clustering Methods, chapter 2.

Figure 4-1

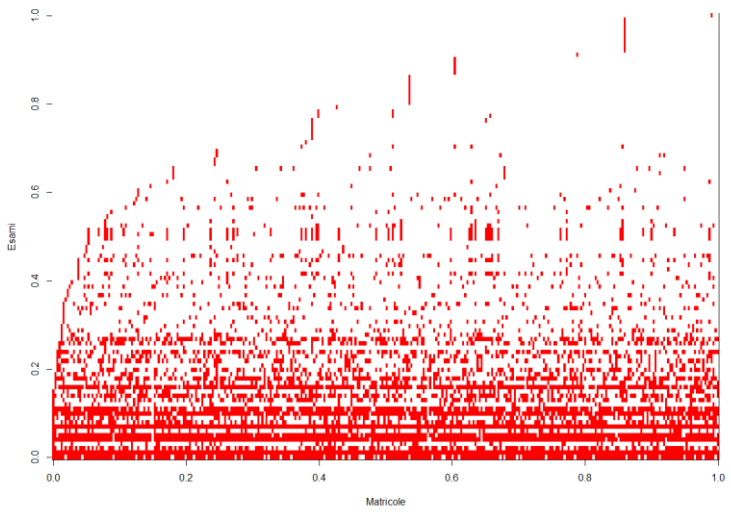


Table 4-2

`binarydataf`	The input binary data frame on which the analysis should be performed.
`technique`	An object of class *list* which specifies which clustering methods should be performed. Valid arguments are: `'k-means', 'k-medoids', 'ward', 'prob'`.
`nclust`	Number of expected clusters. Might be also a vector, in case it is not still known.
`nsim`	Number of times the algorithms of k-means and mixture-model should be repeated. Based on this number, the whole procedure might take several minutes to complete.

4.3.2 Inside-cluster exam frequencies

Let's say a good clustering classification has been found. Function `mostFollowed_byclust(…)` has been provided in case we would like to understand the frequency of each exam inside each cluster found. This can be useful to understand which courses and exams students tend to follow inside each cluster. A list of the arguments it takes is reported in Table 4-3.
Function return a data frame with *v* rows, as the number of exams, and *k* columns, one for each cluster, containing the exam frequencies. This function was used to make different visualization seen in the previous chapter.

Table 4-3

`clustergroup`	A vector of length *n*, as the total of students, containing the cluster assignment for each student.
`X`	The input binary data frame on which the analysis should be performed.
`verbose`	If TRUE, as default, function prints frequencies for the most-folloed exams in each cluster found.
`percentage`	Minimum required frequency to be printed for an exam in *k* cluster. It does not affect the data frame returned, but only the printed part of the function.
`graph`	If TRUE, it returns a histogram showing dimension of each cluster.

Conclusions

Analysis was performed on a university dataset containing information on every single course followed by each graduated student in MSc Informatic Engineering. From that, with the help of domain knowledge, a binary dataset summarizing each student career was build. Various unsupervised analysis techniques were applied to look for any cluster of students in data. The results of those techniques lead to three clusters. One of the three was composed by students that enrolled university before 2012. The remaining two main clusters were identified as *Artificial Intelligence* and *Systems and Networks*, based on the exams frequencies inside them.

References

Hennig, C. (2007). Cluster-wise assessment of cluster stability. In *Computational Statistics and Data Analysis 52* (pp. 258-271). Elsevier.

Hennig, C. (2008). Dissolution point and isolation robustness: robustness criteria for general cluster analysis methods. In *Journal of Multivariate Analysis 99* (pp. 1154-1176).

Ian H. Witten, E. F. (2011). *Data Mining: practial machine learning tools and techniques. 3rd edition.* Morgan Kaufmann.

Kaufman, L. a. (1990). *Finding Groups in Data: An Introduction.* New York: Wiley.

Maaten, L. v. (n.d.). *Bayesian Mixtures of Bernoulli Distributions.* Retrieved from https://lvdmaaten.github.io/publications/papers/TR_BMBD_2010.pdf

Stutz, J., & Cheeseman, P. (1996). Autoclass - A Bayesian Approach To Classification. In J. Skilling, & S. Sibisi, *Maximum Entropy and Bayesian Methods* (pp. 117-126). Kluwer Academic Publishers.

Wikipedia. (2017). *Curse of dimensionality*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Curse_of_dimensionality#Machine_learning

Wikipedia. (2017). *k-medoids*. Retrieved from Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/K-medoids