DATA MINING I PROJECT

PREDICTING EMPLOYEES' ATTRITION: A STUDY

Ludovica Binetti
l.binetti@studenti.unipi.it

Lorenzo Di Gianvittorio
l.digianvittorio@studenti.unipi.it

Stefano Moscatelli
s.moscatelli2@studenti.unipi.it

Lucia Pifferi
l.pifferi@studenti.unipi.it

Digital Humanities, Department of Philology, Literature and Linguistics

Academic Year: 2020/2021

January 6, 2021

ABSTRACT

This report aims to explore and analyze the dataset provided for Data mining 1 course. We would like to show how this dataset can be used to train different types of models that are able to predict the attrition value of a given employee.

1. Data understanding

The dataset is a fictional dataset created by IBM data scientists in order to understand what factors lead to employee attrition in a company. As we will illustrate more in detail in the next paragraph, it is a relatively small dataset consisting of many variables and a reasonable number of records.

1.1 Data semantics and distributions

The dataset gathers information on 1176 employees of a company. For each record there are 33 attributes of different kinds (both binary and non-binary categorical variables and both discrete and continuous numerical ones) with many missing values. The dataset seems more skewed on the number of variables rather than on the quantity of the records, which is why a thorough preemptive analysis is required.
In order to analyze categorical variables and to show their frequency distributions, we used bar plots. The dimensions on which we focused the most in the preprocessing phase are the following:

- Attrition: it is a binary variable (Yes = 192 / No = 984) that describes the attrition of an employee from the company. It is a central dimension as it represents the goal of the whole analysis.
- BusinessTravel: variable that shows how frequently an employee travels for work. It has 3 possible values: Travel_Rarely; Travel_Frequently; Non-Travel. The overall distribution suggests a tendency to travel rarely.
- EducationField: this variable represents the person's field of education. It has 6 values: Human Resources, Life Sciences, Marketing, Medical, Technical Degree and Other. Most of the employees

concentrate on Life Sciences (489 record) and Medical (370 record).

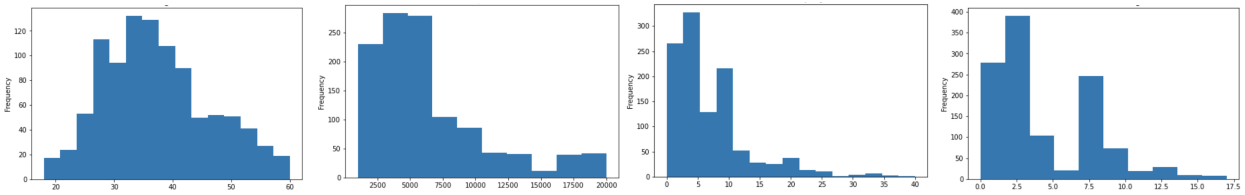
- Over18: it is a binary variable (Yes/No) whose meaning is a bit unclear. It could refer either to the fact that an employee is older than 18 years (which sounds quite obvious) or to the fact that he/she has worked more than 18 years for the company or on his/her whole life. In our opinion, the latter interpretation should be considered. However, whatever meaning it holds, only the 'Yes' value is detected in the dataset.
- PerformanceRating: it expresses the judgment regarding an employee's working performance. Although the range of possible values goes from 1 to 4, only values expressing a positive judgment (3 and 4) can be found in the dataset.
- EnvironmentSatisfaction: this variable describes how satisfied an employee is with regard to the working environment of the company. The satisfaction is expressed on a scale going from 1 (minimum value) to 4 (maximum value). Overall, the records show high satisfaction values (717 records between 3 and 4).
- Gender: it is a binary variable (Male/Female) giving information about the sex of an employee. Most of the employees are men (664 records).

In order to analyze numerical attributes and easily capture their distribution (especially normal ones), we used histograms. The dimensions on which we concentrated the most in the preprocessing phase are the following:

- Age: discrete attribute indicating employees' age. The distribution approximates a normal curve. (Figure: 1.1a).
- MonthlyIncome: discrete attribute representing the salary received by an employee every month. The distribution is denser in the left part, meaning that very few people earn lots of money. (Figure: 1.1b).
- StandardHours: number of working hours. Since it is a "standard" number of hours, only one value (equal for all the records) can be found in the dataset.
- TrainingTimesLastYear: discrete attribute that stands for the number of times (in the previous year) an employee took part in a training program.
- YearsAtCompany: discrete attribute that indicates how long an employee has been working for the company. The distribution is not symmetric, but strongly skewed to the right. (Figure: 1.1c).
- HourlyRate, DailyRate, MonthlyRate: even though their meanings and their reciprocal relations are not completely clear to us, we suppose that these three variables are all salary-related as they indicate how much an employee earns hourly, daily or monthly.
- NumCompaniesWorked: discrete attribute that represents the total number of companies in which a person worked.
- TotalWorkingYears: discrete attribute indicating how many years a person has worked for in his/her whole life. It also includes the years an employee has worked in the company. By looking at the distribution, we see that only few people have worked for many years.
- YearsSinceLastPromotion: a discrete attribute (whose range goes from 0 to 15) that indicates how many years passed since an employee was last promoted. The distribution presents a long tail on the right.

- YearsWithCurrentManager: a discrete attribute that shows how long an employee has had the same manager for. The distribution seems bimodal, with peaks in the values of 2.5 and 7.5 years. (Figure: 1.1d).

- YearsInCurrentRole: discrete ordinal attribute that expresses the number of years an employee has been holding a certain position within the company.



(a) Age **(b)** MonthlyIncome **(c)** YearsAtCompany **(d)** YearsWithCurrManager

Figure 1.1: Histograms of some numerical attributes

The only significant correlations (greater than 0.5) that we notice in the original dataset are between the following couples of variables: JobLevel and TotalWorkingYears; MonthlyIncome and YearsAtCompany; YearsInCurrentRole with YearsSinceLastPromotion and YearsWithCurrManager; YearsSinceLastPromotion and YearsWithCurrManager.

1.2 Data quality

1.2.1 Outliers and missing values

"There is no excuse for failing
to plot and look"

John Wilder Tukey

Before dealing with the missing values, we first identified and removed the outliers from each dimension in order to check if the elimination of those values could lead to significant change in the distributions. The heuristic used for detecting outliers was mostly the IQR * 1.5, but we also inspected visually the presence or absence of outliers using boxplots.
Once the dataset was purged from outliers, we plotted every distribution. The only one approximating a Gaussian distribution was Age. This is why we decided to replace its missing values (in total 176) with the mean. By doing so, the distribution didn't change significantly: as predicted, it boosted only the central peak.
For TrainingTimesLastYear, we grouped by Department and took their individual mode; then we filled accordingly.
Similarly, we filled YearsAtCompany missing values (in total 60). As this variable wasn't correlated with any other feature, we designed a solution that involved grouping the employees in bins of 10 years age-based and calculating the mean for each of them. Then we filled the missing values accordingly.
For dealing with Gender missing values (in total 59), we randomly selected a value between Male or Female, but still respecting the original population ratio between the two of them. The choice of refilling these missing fields was quite harsh since we reckon that the lack of some values could be motivated by someone's unwillingness of disclosing their gender.
Then, we had to deal with MonthlyIncome missing values (in total 213). Since the distribution was not normal, we couldn't exploit the mean as we did for Age. Instead, we checked its correlation with other variables and we actually found out it was correlated with YearsAtCompany (Pearson = 0.513). After building

a regression model, we could fill the values by solving the following equation:

$$y = 391.4x + 3763.8 \tag{1}$$

Finally, we replaced BusinessTravel missing values (in total 107). First, we run a Chi square test to see if there was any significant correlation with another categorical variable in the set. Unfortunately, all the p-values were greater than 0.05, so we had to accept the null hypothesis, by which the variable was independent. Therefore, we decided to substitute the missing values with the most frequent one.

1.2.2 Other data quality considerations

We also would like to stress the presence of the following variables with particular quality characteristics that led us to the decision of not replacing their missing values:

- PerformanceRating: the number of missing values is almost the 10% of our whole dataset (in total 138); as the range of possible values goes from 1 to 4, but in the dataset just the last two values (3-4) appears, there is no reliable way in which we can try to estimate them.
- StandardHours: there are 570 missing values (almost the half of our records); furthermore, the only value that appears is 80. We reckon that this value is incorrect as 80 hours is definitely too much for working either in a day or even a week and too little for working in a month. If we assumed the presence of a format error (eg. it should be 8), still it would be irrelevant.

1.3 Fixing semantic inconsistencies

Once purged the dataset from noise, we calculated the correlation in this new dataset in order to see whether we had an improvement with respect to the previous correlation values, but we had no particular luck.
In reality, we noticed that variables that should have been "semantically correlated" (like YearsAtCompany and TotalWorkingYears) were not because there were lots of inconsistencies in our dataset. Intuitively, the longer people work for a company, the longer they will work in total in their life, but in our dataset some employees had worked in the company more years than they did in their entire life. This could not happen as we assumed that TotalWorkingYears included YearsAtCompany! This is the reason why we decided to correct this and other "semantic errors" by performing some controls as we illustrate below:

- Age and TotalWorkingYears: we assumed that a person could not have started working in early age (before being 18 years old); thus, TotalWorkingYears + 18 must always be smaller or equal to Age.
- YearsAtCompany and TotalWorkingYears: the total working years at company must be equal or smaller than TotalWorkingYears; if not, we substitute the value of TotalWorkingYears with YearsAtCompany.
- YearsWithCurrManager: we assumed that a person couldn't have had the same manager in different companies; so if a person had the same manager for more years than the years spent in the company, we had to replace the value of YearsWithCurrManager with YearsAtCompany.
- YearsSinceLastPromotion: it can't never happen that YearsSinceLastPromotion is greater than YearsAtCompany; if this happended, we made YearsSinceLastPromotion equal to YearsAtCompany .
- YearsInCurrentRole: it's not possible that a person has been holding a position for more years than he/she actually worked for the company (because we previously assumed that the variable expresses the number of years in current role within the company); if this happened, we replaced YearsInCurrentRole value with YearsAtCompany one.

1.4 Data Transformation

We decided to transform our dataset variables in different ways. First of all, we decided to transform the type of all numerical attributes from float to int because they were all round values. Then, we decided to perform binarization of the following categorical attributes:

- Gender: 0 = man; 1 = woman;
- Attrition: 0 = no; 1 = yes;
- OverTime: 0 = no; 1 = yes.

We finally chose to eliminate PerformanceRating and StandardHours. There was no point in keeping these two variables for the reasons previously showed in the data quality assessment.
In this phase of the analysis, we also decided to delete the three dimensions regarding the rates because of their obscure semantics: HourlyRate, DailyRate and MonthlyRate. If MonthlyRate referred to gross income, there would be severe inconsistencies with respect to MonthlyIncome; if it represented, instead, whatever kind of ratio between working time and salary earned, it would still not be consistent with the other variables (which are HourlyRate and DailyRate). The best guess was to assume that MonthlyRate was a kind of value related to earnings, but, since we had already taken into account MonthlyIncome, it resulted a redundant variable. We, therefore, eliminated it with its two "semantic" children.
Over18 was deleted as well, as it didn't provide any useful information for the reasons shown above. Also, as we noticed that JobLevel and TotalWorkingYears were both correlated to each other and exactly to the same variables in the dataset, we decided to eliminate JobLevel.

1.4.1 Creation of a new variable: Stagnation

Since YearsInCurrentRole and YearsSinceLastPromotion carried a very similar meaning and had a strong correlation (0.63), we decided to combine them in a unique variable that we called Stagnation. Assuming that a person can also be promoted while remaining in the same role, we consider Stagnation as the result of the product between YearsInCurrentRole and YearsSinceLastPromotion:

$$Stagnation = YearsInCurrentRole \cdot YearsSinceLastPromotion \tag{2}$$

Multiplying the two variables in this way gives us an idea of the employees' "labor mobility".
For example, a person who has worked in the same role for five years (but has been promoted in the current year) has a Stagnation value equal to 0. So, the variable YearsSinceLastPromotion becomes the weight of the mobility: the lower its values are, the lower Stagnation will be.

1.5 Final dataset overview

At the end of the data understanding and transformation part, we obtained a smaller dataset with 25 dimensions and 1013 records. Figure 1.2 shows the new correlations. Even though some variables are still highly correlated to each other, we decided to keep them since we think they might bring more significant meanings in further steps of analysis rather than they would do individually.

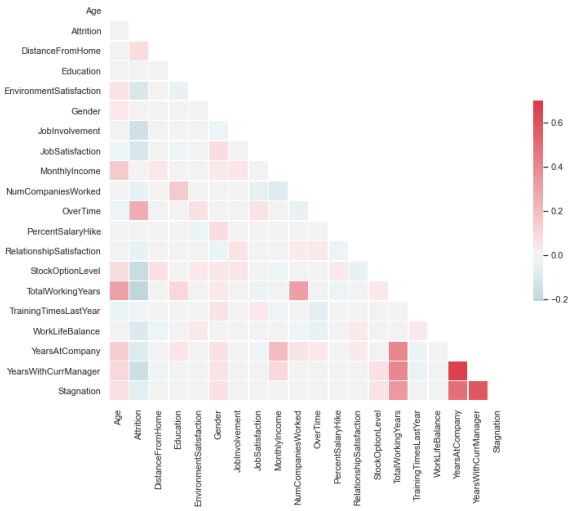


Figure 1.2: Correlation matrix

2. Clustering

After assessing the nature and the quality of our dataset, we proceeded to determine its clusters by implementing three clustering algorithms: Kmeans, DBSCAN and hierarchical agglomerative clustering. The goal is to assess the best way of identifying groups that share common regularities in our data.

2.1 Kmeans

We first preprocessed our data by normalizing the values with the StandardScaler function in order to bring them all under the same scale. We then selected the 3 best features to feed Kmeans with in order to reduce the "curse dimensionality" effects and improve both efficiency and visualization.
These are the steps that we followed for the feature selection:

1. We visually analyzed the scatter matrix to see whether there were any evident associations between variables.
2. We run a Chi Square test (using the scikit function SelectKBest) for all the numerical attributes in order to find the most informative ones. The results in descending order were MonthlyIncome, Stagnation, YearsWithCurrManager, YearsAtCompany and TotalWorkingYears. For validating our discovery, we looked again at the scatter plots that actually proved some kind of relation between the selected variables.

subset	k	SSE
df1	k = 6	691.74404
df2	k = 6	695.49562
df3	k = 6	705.40803
df4	k = 6	944.28359
df5	k = 6	644.55727

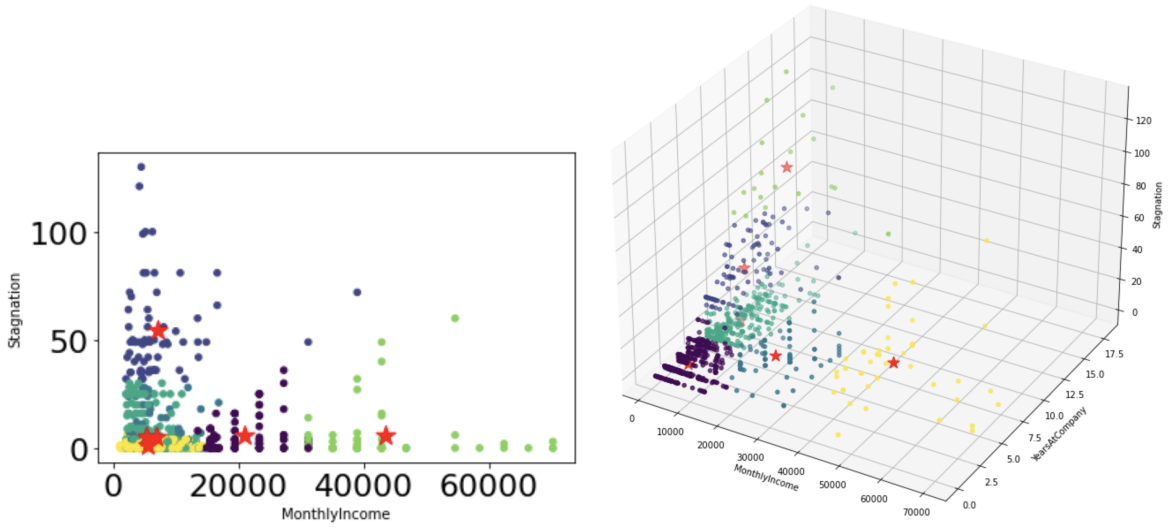
Table 2.1: SSE values for different setups of Kmeans. This table shows the SSE corresponding to each subset of variable provided to Kmeans. We can see that the least error correspond to the subset n° 5 (composed of MonthlyIncome, Stagnation and YearsAtCompany)

We run some tests training Kmeans algorithm with different 3-variables combinations of the top-5 attributes provided by the application of the previous Chi Square test. We eventually came to the conclusion that the best variables to feed Kmeans with were MonthlyIncome, Stagnation, YearsAtCompany. The other parameters for Kmeans have been selected as follows:

- Selection of k: we have found the best trade-off (k = 6) by implementing the Knee method and visualizing the graph that plots increasing values of SSE. The euclidean distance metric has not been modified. A summary of the procedure is shown in table 2.1.
- Number of iterations: we left the default value because even if we tried to increase the number of iterations (n_init > 100), the results returned by the clustering algorithm were pretty much the same as leaving n_init = 10 (default value).

Cluster	MonthlyIncome	TotalWorkingYears	YearsAtCompany	YearsWithCurrManager	Stagnation
1	20905,27	9,49	4,07	2,81	6,00
2	7129,40	13,94	9,24	6,86	54,24
3	6779,00	12,24	9,44	4,57	5,02
4	5132,99	9,10	5,08	2,95	5,51
5	43482,85	8,85	7,02	3,02	5,70
6	5365,49	7,19	1,38	0,96	0,94

Table 2.2: The table shows that each cluster found by Kmeans has specific characteristics regarding the mean of each of the attributes reported in the table.



(a) 2D clustering

(b) 3D clustering

Figure 2.1: 2D and 3D visualization of Kmeans clustering. The 3-dimensional graph clarifies the "closeness" of the bottom-left centroids in the 2-dimensional graph: they are actually "spread" along YearsAtCompany dimension.

The value of the silhouette coefficient is 0.43006, which is average. In conclusion, as we may notice from the figure 2.1a, in the bi-dimensional graph there are 3 centroids that apparently seem similar. Nevertheless, the three-dimensional graph in 2.1b highlights that the grouping is actually appropriate to the centroids found because it develops in width. Finally, as the table 2.2 shows, most of the variations between the identified clusters interest the variables MonthlyIncome, YearsAtCompany, Stagnation (not surprisingly, as they are the variables used to train the model), TotalWorkingYears and YearsWithCurrManager.

2.2 DBSCAN

DBSCAN is the second clustering algorithm that we used. With regard to the feature selection, we decided to feed it with the same features previously used for Kmeans.

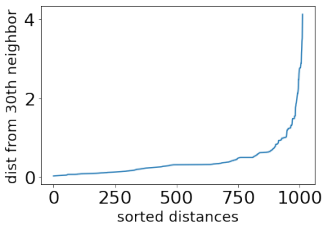


Figure 2.2: Distance from the 30th neighbor measured by SSE

It was not difficult to find the best values for eps and min_values parameters because, after tinkering with them, we obtained two extreme situations: either one huge cluster or very small clusters of few records each. The best compromise between these two situations was to set up eps = 0.22 e min_samples = 30.
The number of clusters was obtained empirically as well, through a for-loop that, at each iteration, increased the number of neighbor points by also setting up different values of epsilon between 0.2 and 1. It was found that, when starting from 30 neighbors, the number of clusters tended to consolidate to 1 in almost all cases, so it was chosen as a threshold value. Once fixed the number of neighbor points, we performed the Knee test to determine a balanced eps value. However, because of the high density of data previously discussed, the optimal value (0.22) was not found in the inflection point but in correspondence of the first significant increase of the 30th nearest neighbor (see fig. 2.2). The chosen distance function was again the Euclidean distance; we actually tried to select other types of distances, but the DBSCAN output would either worsen the quality

of the results or provide little difference.
Hence, after fitting the model, we obtained 7 clusters (including the noisy points) with the following partitions: -1: 536, 0: 193, 1: 57, 2: 53, 3: 74, 4: 45, 5: 55. The clusters are visible in the graph at figure 2.3 and summarized in the table 2.3. The value of silhouette coefficient is 0.00607, which is actually quite low.

Cluster	MonthlyIncome	TotalWorkingYears	YearsAtCompany	YearsWithCurrManager	Stagnation
1	4911,48	6,88	1,00	0,65	0,38
2	3968,37	7,89	3,00	1,82	1,67
3	3231,42	8,40	2,00	1,66	2,42
4	4553,35	8,26	5,00	2,23	1,39
5	3764,00	8,29	4,00	2,62	1,49
6	4070,80	5,84	0,00	0,00	0,00

Table 2.3: Means by cluster according to DBSCAN (noise is not shown)

Surprisingly, DBSCAN finds as many clusters as Kmeans, but smaller. What's interesting is that this approach considers noise everything which is not in the bottom-left quadrant of the graph: 5 out of 6 clusters are all described on YearsAtCompany dimension (in fact the means in the table are round values). DBSCAN tends to ignore all the sparse and extreme values (which are more than half of the dataset): as a consequence, each cluster has lower mean values for each of the variables considered. Finally, DBSCAN seems to have clustered the new employees, whose caracheristics are having 0 in YearsAtCompany, YearsWithCurrentManager and Stagnation fields.

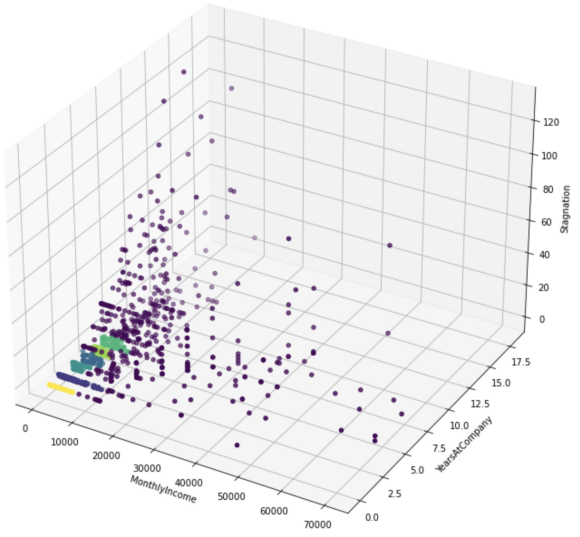


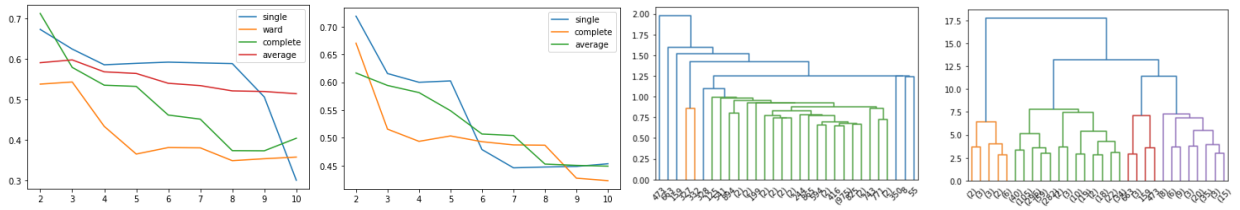
Figure 2.3: 3D visualization of DBSCAN clustering

2.3 Hierarchical clustering

The hierarchical agglomerative clustering is the third and last method we used in this clustering section.
We provided the algorithm the same three variables as before (MonthlyIncome, Stagnation, YearsAtCompany) normalized with the StandardScaler and we run different experiments by mixing linking methods (single, complete, average, Ward) and distance metrics (Euclidean and Manhattan).
Here we mostly focus on single and complete-linkage methods because we think they provide the most

relevant results. After plotting silhouette values and the number of clusters on parallel coordinate graphs for each distance metrics, we found out that:

- The silhouette values of Euclidean distance decrease more slowly with the increase of the number of clusters.
- In the Euclidean distance graph, one of the best silhouette value is in correspondence of single-linking method and k = 8.
- The silhouette values of Manhattan distance decrease more quickly with the increase in the number of clusters (in particular they decrease significantly after k = 6 and k = 7).



(a) Euclidean distance **(b)** Manhattan distance **(c)** MIN + Euclidean dist **(d)** MAX + Manhattan dist

Figure 2.4: Parallel coordinate graphs and dendrograms

After this first overview (where the Euclidean distance seemed the best metric), we analyzed the dendrograms. In the single-linking one, both with Manhattan and Euclidean distance, the silhouette coefficient appeared pretty good ($\simeq 0.6$). However, when we looked at the single-linking dendrogram with Euclidean distance, we noticed that the records were mainly grouped in one big cluster (e.g.: for k = 2, Cluster 0: 1012, 1: 1, Silhouette Score 0.67; for k = 3, Cluster 0: 1011, 1: 1, 2: 1, Silhouette Score 0.62). We reckon that this setup wouldn't have been really informative for our purposes because we wouldn't have been able to identify any significant difference between the clusters identified. We assume that the reason behind this result is the presence of noise which single-linking algorithm is not able to classify correctly.
Instead, when we looked at the complete-linking dendrogram with Manhattan distance, we could spot 3 or 4 clusters. We are aware that the merge of the clusters happens higher than in the previous graph (which means that the records are less similar between each other), but, when we analyzed the cardinality of the groups, we found a slightly more balanced clustering (e.g.: for k = 3, Cluster 0: 105, 1: 16, 2: 892, Silhouette Score 0.51; for k = 4, Cluster 0: 892, 1: 6, 2: 99, 3: 16, Silhouette Score 0.49).
For this reason, we think that the latter method gives a more informative output than the single-linking with Euclidean distance one.

2.4 Final considerations

In order to evaluate the correctness of a clustering method, we have taken into account the silhouette coefficient for every algorithm. Based on this metric, the best clustering is the one using complete-linkage and Manhattan distance (silhouette = 0.49 and k = 4). However, if we consider the cardinality of each cluster that the hierarchical algorithm finds, we observe that most of the records fall inside the first one (Table 2.4), resulting in a very generic grouping.
In other words, the hierarchical algorithm doesn't seem to properly explain the variation of our data. This is why we decided to have a closer look at the mean of some attributes and we observed the following:

- As we have already stated in paragraph 2.2, we don't notice great variations between DBSCAN clusters, probably because they are very close in space. Also, by grouping the employees according to

Cluster	N° di record
5	495
2	271
4	90
0	87
3	54
1	16

Cluster	N° di record
-1	536
0	193
3	74
1	57
5	55
2	53
4	45

Cluster	N° di record
0	892
2	99
3	16
1	6

Table 2.4: Kmeans, DBSCAN, Hierarchical cardinality

YearsAtCompany, the algorithm doesn't take into account the older employees. For these reasons, we decided to discard it.

- From Hierarchical and Kmeans comparison, some interesting variations in data emerge. We have already said that the limit of the hierarchical method is to create a really big cluster and 3 little clusters with very few data. Moreover, we would like to stress the fact that Kmeans finds more clusters than DBSCAN but, still, its silhouette score is just a little smaller than DBSCAN one (0.43 vs 0.49).

We can therefore conclude that the best clustering algorithm is Kmeans because it finds the most informative clusters in our dataset.

Cluster	MonthlyIncome	TotalWorkingYears	YearsAtCompany	YearsWithCurrManager	Stagnation
1	5977,38	8,83	4,02	2,40	5,32
2	44207,67	12,17	10,00	8,17	46,00
3	33690,79	9,20	5,51	2,83	5,69
4	7867,13	15,94	11,88	7,50	84,00

Table 2.5: Means by cluster according to HAC

3. Classification

Before performing classification, we modified and transformed the test set (which was provided in a separate file) following the same steps taken in the Data Understanding section, except from elimination of outliers, correction of semantic inconsistencies and binarization of the variable Attrition. Modifying the test set was mandatory as the two sets - training and test - were not equal anymore because we eliminated redundant variables, transformed some of them and added a new one (Stagnation). Not fixing these issues would have led, of course, to inconsistencies in classification.

3.1 Decision tree

Since the algorithm implemented in scikit-learn didn't allow categorical variables, we preprocessed the data with the Label encoder, mapping the values of our categorical variables to discrete numbers. After this, we used two selection models (RandomSearch and GridSearch) in order to perform the parameter tuning of the decision tree. Once found the best possible sets of parameters, we decided to do model selection with cross-validation (k-fold with k = 10), evaluating, for every trained model, the accuracy and F1 score, both calculated on the validation set. Below, we show the results of the three trained models:

	Precision	Recall	F-1 score	Support
No	0.88	0.87	0.88	249
Yes	0.33	0.36	0.34	45
Accuracy			0.79	294
Macro avg	0.61	0.61	0.61	294
Weighted avg	0.80	0.79	0.80	294

Table 3.1: Scores according to the best Decision Tree model

1. clf = gini, max depth = None, min samples split = 5, min samples leaf = 1. Accuracy: 0.7434 (+/- 0.18) F1-score: 0.6005 (+/- 0.12).
2. clf = gini, max depth = 10, min samples split = 2, min samples leaf = 1. Accuracy: 0.7364 (+/- 0.18) F1-score: 0.6041 (+/- 0.14).
3. clf = gini, max depth = None, min samples split = 10, min samples leaf = 20. Accuracy: 0.8094 (+/- 0.08) F1-score: 0.5706 (+/- 0.09).

Among the three, the first model is definitely the best one and we can see that the gain metric entropy is never taken into account. Even if the third decision tree seems the most accurate one, it is clear that the low F1-score shows a non-negligible presence of false positives and false negatives. Therefore, we prefer a model with a higher F1-score but a slightly lower accuracy in spotting the true positives and true negatives. After establishing the best model, we used it to predict the values of attrition on the test set, obtaining the metrics showed in table 3.1. What we can see is an acceptable score in the prediction of the 'No' label (with 223 record classified correctly), followed by a quite poor accuracy on the 'Yes' label, which scattered its values among false positives and false negatives (to be more precise, 30 values over 45). Certainly, it has to be considered the few numbers of records leading to the 'Yes' label (45 records labeled as 'Yes' vs 249 labeled as 'No'), which could be theoretically balanced by a weighted confusion matrix that would assign more weight to the false negatives. However, the results on the test set reveal an accuracy of 0.79 and a F1-score of 0.88 for 'No' and 0.34 for 'Yes'.
Finally, the ROC curve (figure 3.1) shows that there is no way to obtain a good precision on the true positives without including, at the same time, a good amount of false positives. We believe that the training set might need a bigger number of records labeled as 'Yes' to be able to achieve higher performances during the testing phase.

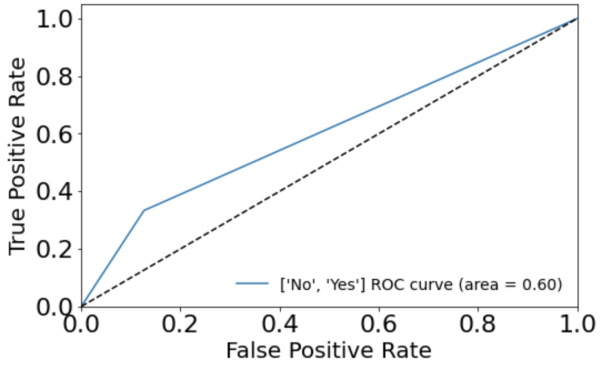


Figure 3.1: ROC curve graph

DT

	precision	recall	f1-score	support
No	0.88	0.87	0.88	249
Yes	0.33	0.36	0.34	45
accuracy			0.79	294
macro avg	0.61	0.61	0.61	294
weighted avg	0.80	0.79	0.80	294

Table 3.2: Summary of performance measures of DT

KNN

	precision	recall	f1-score	support
No	0.85	0.87	0.86	249
Yes	0.16	0.13	0.14	45
accuracy			0.76	294
macro avg	0.50	0.50	0.50	294
weighted avg	0.80	0.79	0.80	294

Table 3.3: Summary of performance measures of KNN

3.2 KNN

In order to define the best values for the KNN parameter (number of neighbors and weighting score), we used GridSearch. The best results are achieved with k = 1 and uniform weighting.
We are aware that tuning k = 1 is risky because the model could very easily rely on noisy points, but it was the only way to discard very low F1-score values. By running KNN algorithm with k = 1, we obtained 0.72 in accuracy and 0.50 in F1-score on the validation set (the cross-validation was made with k = 10). We, finally, applied our model to the test set, achieving a slightly better result: 0.76 in accuracy, 0.82 in F1-score for 'No' labels and 0.34 in F1-score for 'Yes' ones.

3.3 Final considerations on classification

A common trait of both the classification models is that they share high values for accuracy and precision/recall in the prediction of the 'No' label, while the same values reach the bottom in the prediction of the 'Yes' label. A possible explanation for the low performance lies in the very few examples supporting 'Yes' with respect to those supporting 'No'. Moreover, it may be due to the test set itself, similarly to the training set: in fact the test set contains some semantic inconsistencies that have not been fixed in order to better simulate noise and analyze the behaviour of the model with totally new data. For these reasons, both the algorithms tried to classify some inputs that were not similar to the model they've been fitted on. In conclusion, by looking at the averaged value of the F1-score (compare tables 3.2 and 3.3), we pick Decision Tree as the best classification model.

4. Association Rules

In this section, we will analyze the dataset and we will try to identify the most frequent and interesting patterns. Our final aim is to discover useful association rules that will help us performing some other tasks, like prediction of missing values and of the target variable. In order to "play around" with our dataset, we will apply the apriori algorithm to both the whole dataset and to a subset made with the most informative variables that have been discovered in a previous section of this report (see paragraph 2.1).

4.1 Preprocessing

In order to run the apriori algorithm, we necessarily had to preprocess our dataset by transforming all of our numerical attributes in categorical ones and by binning continuous numerical variables so to reduce the number of possible values. We are now going to illustrate the steps we followed in the preprocessing phase:

- Age: we reduced this variable to 4 bin of equal length;
- Attrition: we brought the 0-1 values in No-Yes values and we also added a string to let us identify the attribute later on;
- DistanceFromHome: we created 3 bins and we assigned to each of them different labels (near_Dist, medium_Dist, far_Dist);
- Education, EnvironmentSatisfaction, JobInvolvement, JobSatisfaction: we converted the values into the correspondant strings provided in the original specification that we found on Kaggle platform.
- Gender: we converted binary values into strings (Male and Female);
- MonthlyIncome: by looking at the variable distribution (figure 4.1), we decided to create 3 bins corresponding respectively to low, medium and high income;
- NumCompWorked: we created 3 bins labeled as 'some_Comp', 'many_Comp' or 'none_Comp'. This last one represents the case in which the value is 0;
- OverTime: we simply added a string that would allow us to immediately recognize the variable in further steps of analysis;
- PercentSalaryHike: as the distribution was unbalanced towards the lowest values, we performed statistical binning creating 2 bins of equal cardinality by splitting the whole interval of values according to the median;
- RelationshipSatisfaction, StockOptionLevel, WorkLifeBalance: we converted values into strings that we arbitrarily decided;
- TotalWorkingYears: we created 3 bins to identify employees in the company with less than 10 years of service, between 10 and 20, and with more than 20 years of service;

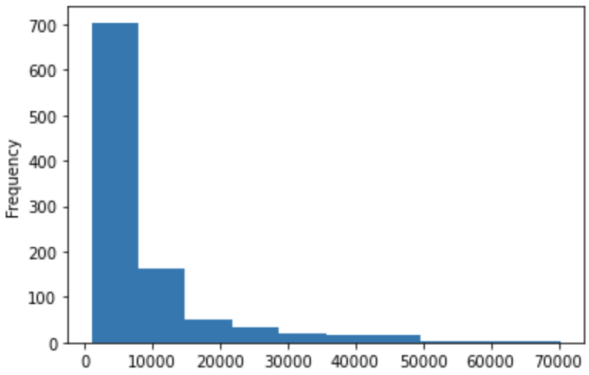


Figure 4.1: MonthlyIncome distribution

- TrainingTimesLastYear: we binarized the variable in order to focus on the presence or the absence of whatever number of times an employee was trained;
- YearsAtCompany: we created 2 bins to identify workers with less than 10 years of service and more than 10 years of service in the actual company;
- YearsWithCurrManager: we created 3 bins of the same width (3 years each);
- Stagnation: in order to identify a binning criteria for this variable, we looked at its distribution. From the histogram 4.2, we observed that most of the records concentrate in the interval from 20 to 50. Therefore, we created 3 bins respectively containing values smaller than 20, between 20 and 50 and greater than 50.

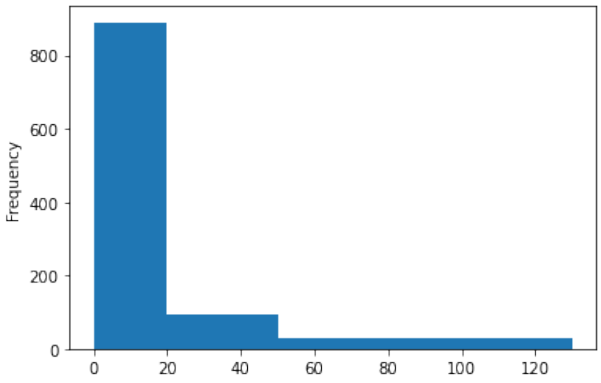


Figure 4.2: Distribution of stagnation

4.2 Frequent Itemsets

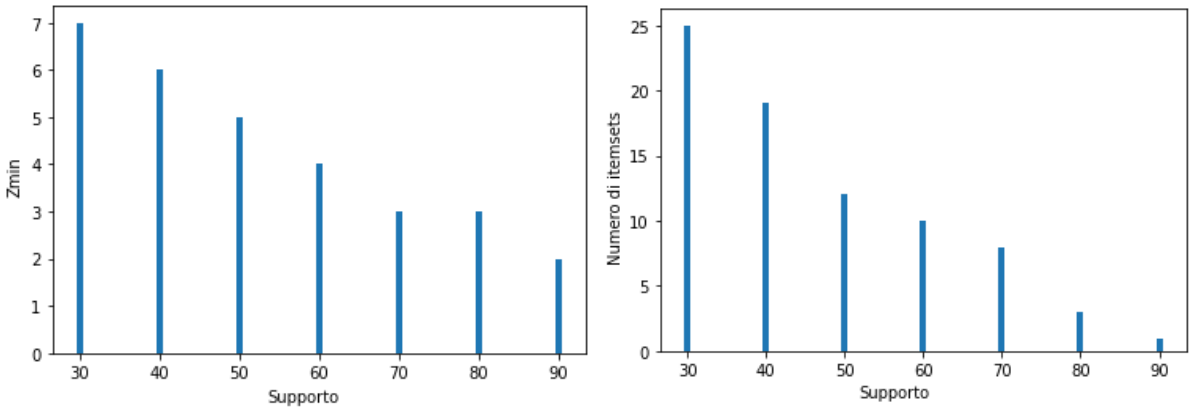
Finding frequent itemsets was relatively easily as all we had to do was running the apriori algorithm. The most challenging thing was actually to tune the function parameters. A good solution for us seemed to run a grid search with different support, zmin parameters and different types of frequent itemsets, and, also, to filter the results according to the number of itemsets found. With a support ranging from 30 to 90, a zmin ranging from 2 to 15 (we decided such a big interval in order to cover as much as possible all the dataset) and considering 'a', 'm' and 'c' as possible types of itemsets, the GridSearch extracted 279 results. These were filtered with a "if condition" that pruned all the combinations whose length was greater than 100 or equal to 0. As we mentioned before, we also split the analysis in two steps: the first step was made on the whole dataset, the second one on a subset based on the most informative attributes. So the GridSearch was actually run twice: one for each step. In both cases, to reduce the results, we chose to find only the maximal itemsets because the GridSearch showed duplicated itemsets whenever a maximal one was found (and this is, obviously, for the definition itself of maximal itemset, which covers also closed and frequent ones). Once run a series of experiments in order to explore all the possible combinations of parameters, we concluded that the best combinations of parameters were the following.
When considering the whole dataset:

- The support was set between 70 and 90.
- Zmin was decided to be equal either to 2 or 3. We actually wanted to consider higher values of zmin as we considered lots of variables. But, because of our choice to select quite high support values, the Grid Search wouldn't show any other zmin values apart the two selected.

- As report, we decided to show the percentage of relative item set support.

When considering the sub-dataset (that had 10 variables):

- The support was set between 70 and 90
- The zmin range varied from 2 to 10. However, we noticed not only that the zmin parameter wouldn't go further than 7 but also that it tended to decrease when the support increased. The graphs 4.3a and 4.3b show that, as the support becomes bigger and bigger, both the zmin parameter and the number of found itemsets tend to decrease.
- As report, we decided, again, to show the percentage of relative item set support.



(a) Zmin variation

(b) Number of itemsets variation

Figure 4.3: As the support increases both zmin parameter and number of itemsets decrease

What we actually discovered is that the frequent patterns found by apriori algorithms were pretty much the same in both cases. For example, all of the following patterns were extracted in both cases:

- if support = 80 and zmin = 2:
  - item set 1: No_Attr, less 10 years_YAC - relative support = 80.35%
  - item set 2: No_Attr, yes_TrainTimesLY - relative support = 80.65%
  - item set 3: low_Stag, less 10 years_YAC, yes_TrainTimesLY - relative support = 84.1%
- if support = 80 and zmin = 3:
  - item set 1: low_Stag, less 10 years_YAC, yes_TrainTimesLY - relative support = 84.1%
- if support = 90 and zmin = 2:
  - item set 1: less 10 years_YAC, yes_TrainTimesLY - relative support = 93.97%

As we can see, the third itemset of the firs block (supp = 80 and zmin = 2) is repeated in the second block (supp = 80 and zmin = 3) because the only difference between the two blocks is zmin parameter. We, then, admit that running the apriori algorithm with increasing values of zmin when the support remained the same was actually redundant. If we look at the semantics of these patterns, they actually make sense as they describe expected behaviours in a company: for example, a young employee doesn't suffer from attrition and stagnation.

4.3 Association rules extraction

For this task we actually run, again, the apriori algorithm both on the whole dataset and on a subset of it; but, as we could expect from the the previous section, we ended up with the same results in both of the cases. We therefore acknowledge that making this distinction here would be pointless; instead, we tried to explore a different subset of the dataset as we will explain later. As we did in the frequent itemsets extraction, we run a GridSearch exploring different possible patterns combinations:

- The support ranged from 70 to 90;
- Zmin parameter from 2 to 5;
- The confidence from 70 to 90 (so we aimed for high degrees of confidence).

At the end of this process and after filtering rules by lift (we just took the rules whose lift was less than 0.95 and greater than 1.05 in order to exclude statistically independent). This is what we observed from the extracted rules:

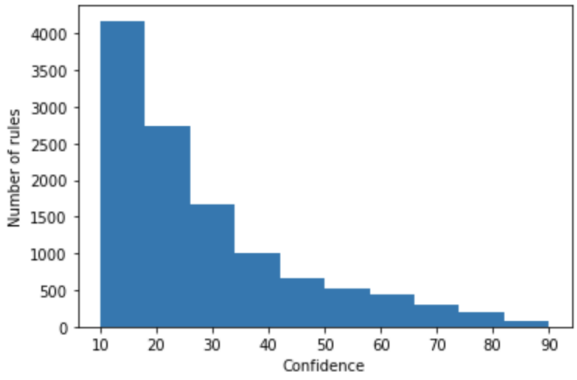
- Continuous numerical variables that have been previously categorized appear seldom or not at all;
- The highest zmin is 4;
- The most frequent variables are: OverTime (only in its 'No' value), Attrition (same as before), TrainingTimesLastYear (only in its 'Yes' value), YearsWithCurrentManager (the bin 'less than 10 years'), Stagnation (the bin labelled as 'low');
- Many rules are redundant. We noticed that this happens because the apriori algorithm often finds either the same rule with a higher degree of confidence or different rules with the same items shuffled. In the former case, we decided to take the rule with the highest confidence.

In the next step of our pattern mining analysis, we analyzed all the rules found and we identified the following as the the most interesting ones:

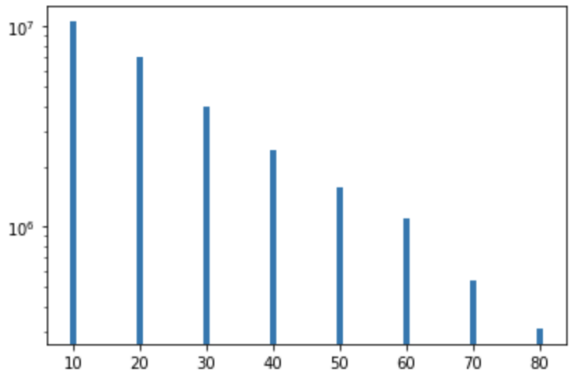
- '0-3 years_YWCM', 'less 10 years_YAC', 'yes_TrainTimesLY' => 'low_Stag'. Confidence: 98,74%, Lift: 1,12;
- 'No_Attr', 'low_Stag', 'less 10 years_YAC' => 'No_OT'. Confidence: 76.67%, Lift: 1.07;
- 'No_Attr', 'low_Stag', 'yes_TrainTimesLY' => 'No_OT'. Confidence: 76.19%, Lift: 1.07
- 'No_OT' => 'No_Attr'. Confidence: 89.44%, Lift: 1.07

Earlier we mentioned that we also run some experiments with a new subset of our dataset: we basically built it out of the subset that has been used in the previous section but adding to it some variables (like BusinessTravel, Departement, EducationField, MaritalStatus, WorkLifeBalance...) that we wished could be taken more into consideration by the algorithm. However, one more time, we didn't find any meaningful difference from the results obtained in the whole dataset. The only way to make other combinations of items and new rules emerge was to abandon the purpose of finding the highest support and confidence rules and to lower the two thresholds. By doing so, we found an interesting rule about the income.

- '[28.0, 38.0)_Age', 'yes_TrainTimesLY' => 'low_Inc'. Confidence: 79.89%, Lift: 1,05;



(a) Variation of the number of rules w.r.t. the confidence parameter



(b) Confidence distribution in logarithm scale

Figure 4.4: Confidence variation graphs

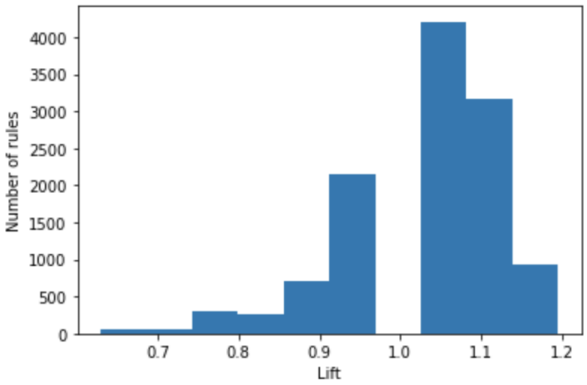


Figure 4.5: Variation of the number of rules w.r.t. the lift

At this point, with the goal to analyze the distribution of the number of extracted rules with respect to the confidence, we run again a GridSearch with a fixed value of support (equal to 70) and zmin equal either to 2 or 3. The only parameter that was changing in a larger interval was the confidence (that ranged from 10 to 90). After doing so, we plotted a histogram (showing the variation in the number of rules w.r.t. the increasing confidence) and a barplot (showing the distribution of the confidence).
We did the same with respect to the lift. The first graph (figure 4.4a) shows a negative correlation between the number of rules and the confidence: by increasing the first the other diminishes. The second graph (figure 4.4b) shows a descending trend in the distribution, which was expected after seeing the previous plot. The third graph (figure 4.5) shows a sort of normal distribution of the lift, except for a total absence of values in the close range of 1 (as we previously filter the association rules).

4.4 Missing values - Over Time

By looking at the variables used in the extracted rules, we chose the binary variable OverTime as dimension to fill some missing values and, therefore, to put in practice the association rules.
From the training set, we randomly selected 240 values (corresponding almost to the 23% of the whole dataset), saved and removed the OverTime value and listed the dimensions of the records. We, then, applied the association rules, which could predict only the 'No_OT' value on 183 record over a otal of 240. This is why, in order to score the accuracy, we didn't consider the 57 records that the rules could not classify. We

also compared the predicted labels to the real ones and we found out that the rules scored an accuracy of 67%. As we don't have any rules that can predict the 'Yes_OT' (plus it is, itself, quite rare in the dataset), we are quite satisfied of the results obtained.

4.5 Target Variable - Attrition

Attrition has always been our main target variable, so we decided to use it one more time in order to test the performance of the association rules. As we wanted to test the effectiveness of the association rules on totally new examples, this time we considered the test set. But of course, before using it, we run all the steps of the preprocessing phase that has been explained in section 4.1. The 22 rules that were found could not classify 138 records (more than 50% of the selected values), which was expected because we didn't completely clean the test set from noise. However, the remaining records where predicted correctly 88.3% of the times. But again, we have to remember that most of the cases had to predict the 'No' value, which is the most frequent label.

5. Conclusions

For sure, the dataset that was provided wasn't easy to deal with. It took us a while to understand the data and we reckon that, in the first part of analysis, fixing semantic inconsistencies was extremely useful because it led to a more consistent dataset. Another thing we would like to stress is the addition of the new variable Stagnation which revealed to be quite useful. In fact, Stagnation ended up being one of the most informative attribute.
Despite the poor information conveyed by the dataset, we must admit that Kmeans gave us not negligible results. Less pleased we are with the results obtained by both DT and KNN in the classification task. As we also mentioned in section 3.3, we think that this might be due to the semantic inconsistencies that we didn't fix in the test set in order to simulate the behaviour of the algorithms with real examples that often contain noise.
Finally, the results of the pattern mining through association rules is unsatisfying. We couldn't determine a way to predict with good coverage in the test set all the values for the target variable (Attrition); moreover, most of the times, the high-confidence rules that predicted the most frequent value ('No') couldn't find the pattern and, therefore, they were unable to assign a label.