IBM HR Analytics Employee Attrition
Data Mining 1 2020/21

Group 23

Alexandra Bradan (530887)[1], Dafin Leva(599842)[2], Eleonora Cocciu (518746)[3], and Marianna Abbattista(621809)[4]

Contacts: [1]a.bradan@studenti.unipi.it, [2]d.leva@studenti.unipi.it, [3]e.cocciu@studenti.unipi.it, [4]m.abbattista1@studenti.unipi.it

February 11, 2021

Contents

1 Data understanding

1.1 Data semantics

The dataset contains information about IBM employees' attrition. It has **1176 records** and **33 features**, as shown in Table 1:

Categorical	Ordinal	Continous
Attrition, BusinessTravel, Department, EducationField, Gender, JobRole, MaritalStatus, Over18, OverTime	Education, EnvironmentSatisfaction, JobInvolvement, JobLevel, JobSatisfaction, PerformanceRating, RelationshipSatisfaction, StockOptionLevel, TrainingTimesLastYear, WorkLifeBalance	Age, DailyRate, DistanceFromHome, HourlyRate, MonthlyIncome, MonthlyRate, NumCompaniesWorked, PercentSalaryHike, StandardHours, TotalWorkingYears, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCurrManager

Table 1: Dataset's features

The majority of the employees present in our data-set are married men, have on average 37 years, are about 9 units far apart from the workplace, in general travel rarely, have a Life Sciences Bachelor Degree, explained due to a greater availability of work positions in the Research&Development department, where they work mainly as level 1-2 Sales Executive, Research Scientist and Laboratory Technicians. They are holding their current job roles on average for 4 years, but every 2 years they are liable to promotions. Their work stability is also safeguarded by managers' shift every 4 years. They have a background of 2 to 3 previous companies and on average a total of 11 working years. Almost all employees aren't leaving the company and in fact they are compelled to stay by granting them on average 1 company's stock and on average 15% of salary hike. Their years at company are on average 7, last year they have undergone a mean of 2 training times, they mainly don't work overtime and as a consequence they gave an high score to job satisfaction and involvement.

With the addition of our three new features:

1. **OverallSatisfaction**: average satisfaction score:
$$\frac{EnvironmentSatis. + JobInv. + JobSatis. + RelationshipSatis. + WorkLifeBalance}{5}$$
2. **TaxRate**: gross Income tax rate:
$$\frac{MonthlyRate - MonthlyIncome}{MonthlyRate}$$
3. **MonthlyHours**: monthly working hours:
$$\frac{MonthlyRate}{HourlyRate}$$

we were able to extract some additional information. For example, we found out that employees' overall satisfaction and involmenent score is medium rather than overall high (this means that there is at least one field where the company could make things better). On average employees work 237 hours per month, an employee is paid on average 14396$ gross income and 6566$ net income, with an average taxation upon the gross income of 45%.

1.2 Attrition's causes

We investigating attrition causes by means of **chi-squared** and **mutual information statistics**, which helped us rank features to use in the clustering process, too. We find out that who leaves usually has spent less than ten years at the company, signaling an internal company uneasiness or the idea to make some change at some point of the career. Employees' dissatisfaction is expressed in Figure 1, where we can notice, for example, how despite scoring an high or a very high score for RelationshipSatisfaction and WorkLifeBalance, three out of our four leaving managers are complaining with the other three scores. In particular, these three managers are all women:

- the first manager assigned a low score to JobInvolvement and to EnvironmentSatisfaction, signaling for hypothesis the existence in the company of a centralized power at the top, which leverage sexual disparity in the Sales department (she is the only female manager in fact here);
- the second manager complains with EnvironmentSatisfaction, too, but in this case, since she is a married and over time worker, probably her complains are against HM's policies;

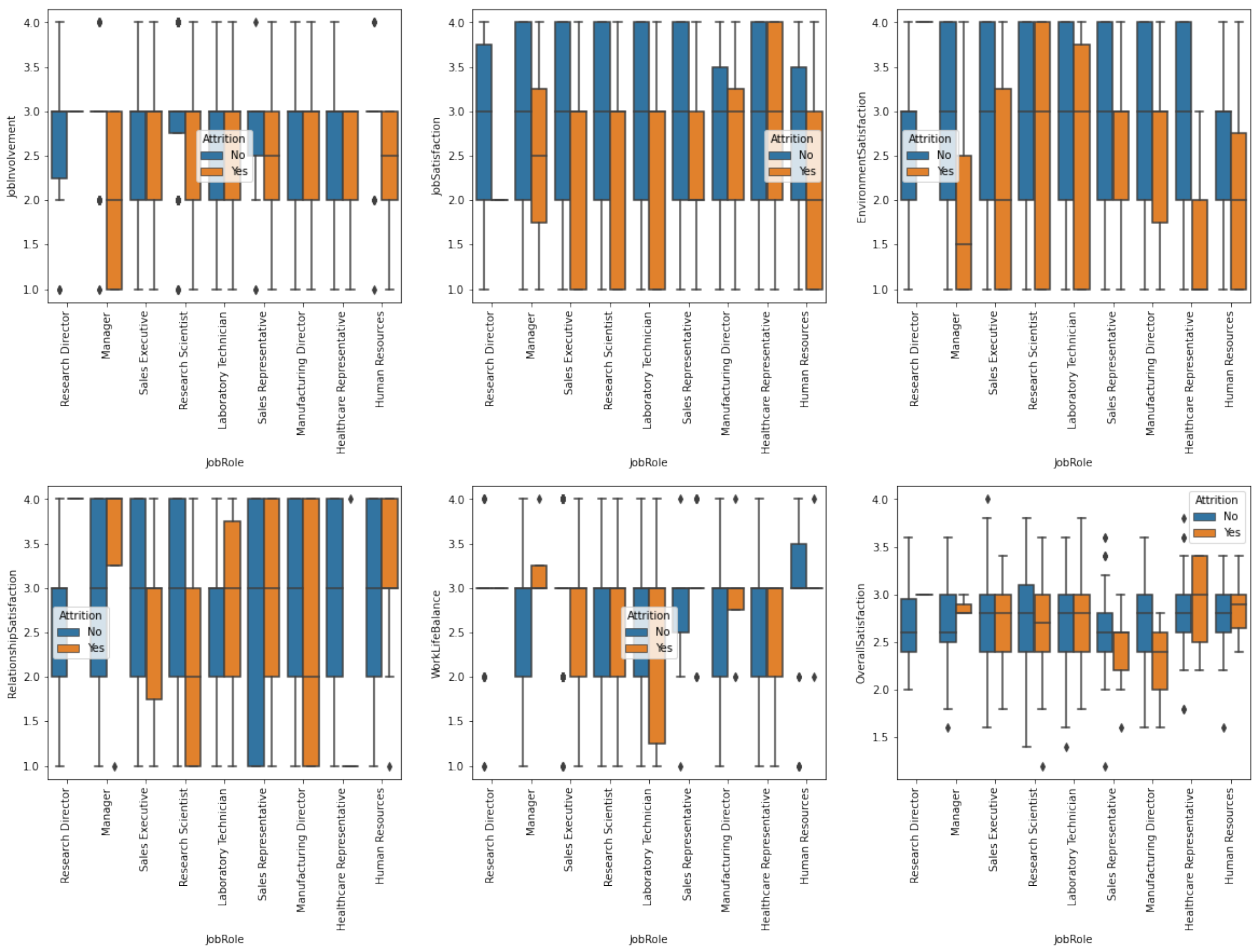


Figure 1: Satisfaction and involvement in leaving employees

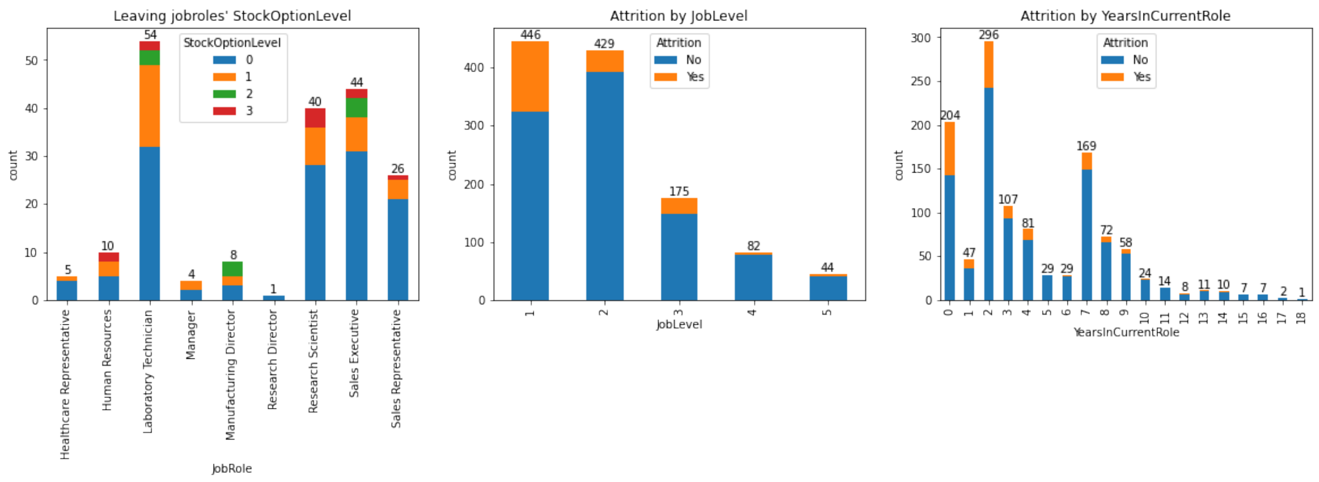


Figure 2: Leaving workers' StockOptionLevel, JobLevel and YearsInCurrentRole

- the third female manager complains for her low fulfillness and happiness in current role, with high regard due to its fresh entering the role among her long-lived in Research & Development colleagues.

  Lastly, the only male manager leaving is unsatisfied on all examined fronts.

The YearsInCurrentRole are lower for the one leaving, as well as having a lower JobLevel, underling how a lower expertise make employees more flexible to leave. A counterpart tendency is expressed by the ownership of company's stock, which compels employees to stay. However there are some exceptions as shown in Figure 2.

Additionaly employees who leave are between 30-40 years, have a Bachelor, have worked at least at a previous company (so we image are quite accustomed to leave), have a low MonthlyIncome in comparison to a greater TaxRate, a lower PercentSalaryHike and usually perform OverTime.

1.3 Data cleaning and dimensional reduction

. We notice that in our dataset some records have unconstrained values, such as rows having:

1. MonthlyIncome < DailyRate;
2. Age - TotalworkingYears < 16;
3. Age - YearsAtCompany < 16;
4. TotalworkingYears < YearsAtCompany;
5. YearsInCurrentRole < YearsSinceLastPromotion;
6. YearsAtCompany < YearsInCurrentRole and NumCompaniesWorked = 0;
7. YearsAtCompany < YearsSinceLastPromotion and NumCompaniesWorked = 0;
8. YearsAtCompany < YearsWithCurrManager;
9. MonthlyRate < MonthlyIncome.

We think the above records are inconsistent, since:

1. MonthlyIncome is the result of DailyRate multiplies by the number of working days in a month, from which is subtracted taxes and additional levies. For this reason ideally MonthlyIncome must be greater than DailyRate;
2. stating IBM's policies [1], the company doesn't use child labor, where the term "child" refers to any employed person under the age of 16;
3. for the same reason of above, an employee can't work in the current company under the age of 16;
4. ideally, an employee's years at company counts his contract working years at the company, so this value has to be smaller or equal the total number of his working years;
5. years since last promotion must be at least equal to the years in current job, smaller if in the promotion years are counted jobrole's advancement levels;
6. if an employee is in his current jobrole more years than he is working in the current company, he must had worked at least in a previous company;
7. if last employee's promotion is elder than his working years at the current company, he must had worked at least in a previous company;
8. an employee's years with current manager must be within his working years at the current company;
9. since MonthlyIncome equals the net income and MonthlyRate equals the gross income, the first must be smaller the second.

We have dropped records which didn't respect 1-8. By taking into account the validity of Age and YearsAtCompany attributes [2], we dropped the following inconsistent features, too:

1. **TotalWorkingYears**: 321 records have YearsAtCompany > TotalWorkingYears;
2. **YearsWithCurrManager**: 361 record have YearsWithCurrManager > YearsAtCompany.

These two features elimination are justified also by their positive correlation with JobLevel(+0.774) and with YearsInCurrentRole (+0.711), respectively.

By taking into account the validity of MonthlyIncome and MonthlyRate features [3], we dropped the following inconsistent additional feature:

[1] https://www.ibm.com/ibm/responsibility/policy11.shtml
[2] All records have Age - YearsAtCompany >= 16
[3] We were able in fact to derive the column TaxRate by their relationship.

1. **DailyRate**: we tried dividing MonthlyRate with DailyRate to found out employee's working days in a month, but we obtained incosistent values for 341 records. We tried dividing DailyRate with HourlyRate to found out employee's working hours in a day, too, but we obtained incosistent values for 130 records and since DailyRate is quite strange with the previous two columns, we prefer to drop it.

For our imposed MonthlyRate $>$ MonthlyIncome constrain, we thought that since the two feature's name start both with "Month" and semantically are quite related, they could had been part of a type error. To establish if the switch among the two features had occurred, we measured neighbours[4]' TaxRate Interquartile Range Method (IQR). Since the IQR can be used to identify outliers by defining limits on the sample values that are a factor k[5] of the IQR below the 25th percentile or above the 75th percentile, our reasoning is that if by computing TaxRate among a record having MonthlyRate $<$ MonthlyIncome (reversing the inequality sign in the TaxRate computation), we get a TaxRate value above or bellow its neighbours' TaxRate IQR, the switched for this particular record hasn't happened and so it must be removed because it don't comply with our imposed MonthlyRate $>$ MonthlyIncome constraint. On the other hand, if TaxRate stayed within neighbours' TaxRate IQR, the record is not a sort of outlier, and so we can proceed to switch its MonthlyRate and MonthlyIncome entries, too (we already computed its TaxRate, using as MonthlyRate the MonthlyIncome and vice versa, to validate the record or not). All out biased tested records, actually, stayed within our imposed cut-off, so we resulted swapping all records' values having MonthlyIncome $>$ MonthlyRate.

We removed additional outlier records applying the Interquartile Range Method (IQR) over the entire dataset, resulting in 883 rows and 32 columns, since we dropped three additional features, too: **Over18** and **StandardHours** were one-valued columns, while **PerformanceRating** had three values (3, 4 and NaN), which are a minigless validation to employees' performance, which results always excellent or outstanding.

1.4 Data imputation

First we checked if numerical attributes might present missing values (i.e. represented by zero values), but all zeros found were meaningful (i.e. 0 YearsAtCompany). The following columns were imputed:

- BusinessTravel, Gender and TrainingTimesLastYear using column's mode;
- YearsAtCompany, MonthlyIncome and Age, using a self-built imputer.

To replace these last sensible features, we opted to guess the missing value by:

1. taking a set of correlated variables (JobRole, JobLevel, YearsAtCompany, StockOptionLevel, YearsInCurrentRole, PercentSalaryHike, Age, MonthlyIncome);
2. searching similar neighbours in the DataFrame, first, by looking at records having the exact values of the features mention above and taking their average values, checking in the mean time our third constrain 3;
3. if the previous constrain wasn't met or none record in the DataFrame shared the same values as the missing one, we started to be more relaxed on the YearsInCurrentRole, YearsAtCompany and Age features, allowing the searching of neighbours having a cut_off=+-5 above or below them. Again we took neighbours average missing value, if it complies with 3;
4. if no neighbour wasn't found again, we sequentially dropped all the other features, until arriving to look after neighbours'records discriminated only by JobRole.

Once replaced the missing values we could normalize our data and look at features correlation, summarized in Figure 3, keeping as well a copy of the cleaned, not normalized data. From the Pearson correlation we found out that, we can dissmiss **MaritalStatus**, **EducationField**, **Department**[6] **YearsSinceLastPromotion**, **HourlyRate** and **MonthlyRate** features which are correlated and so implicitly encode in StockOptionLevel[7], JobRole, JobRole, YearsInCurrentRole, MonthlyHours and TaxRate features. We have also a correlation between our additional OverallSattisfaction feature and the features used to derive them. We will simply not use them together and depending of the granularity of our tasks, we will be more precise or more approximated thanks to them. Our final dataset is made out of **883 records** and **24 features**.

[4]Neighbour's were found as described in Section 1.4, taking extra care in this step, to avoid selecting neighbors having MonthlyRate $<$ MonthlyIncome as the records we are trying to validate.

[5]We used a value of k=3 for our swapped record's validation method, to avoid being to restrictive in the "outlier" (not valid) swapped records. Indeed, we used a value of k=1.5 for the actual outlier removal part, described afterwards.

[6]By removing the Department feature, for all jobroles is unambiguous company's division belonging. The only excpetion is represented by the Manager role, which needs an explicit extension to avoid loosing its department bond. Accordingly, before removing the Department JobRole column, we modified every *Manager* occurrences in the JobRole column, appending the relative Department to it, i.e. *Manager_ Human_ Resources*, *Manager_ Research_ &_ Development*, *Manager_ Sales*.

[7]StockOptionLevel is granted usually to married/divorced employees.

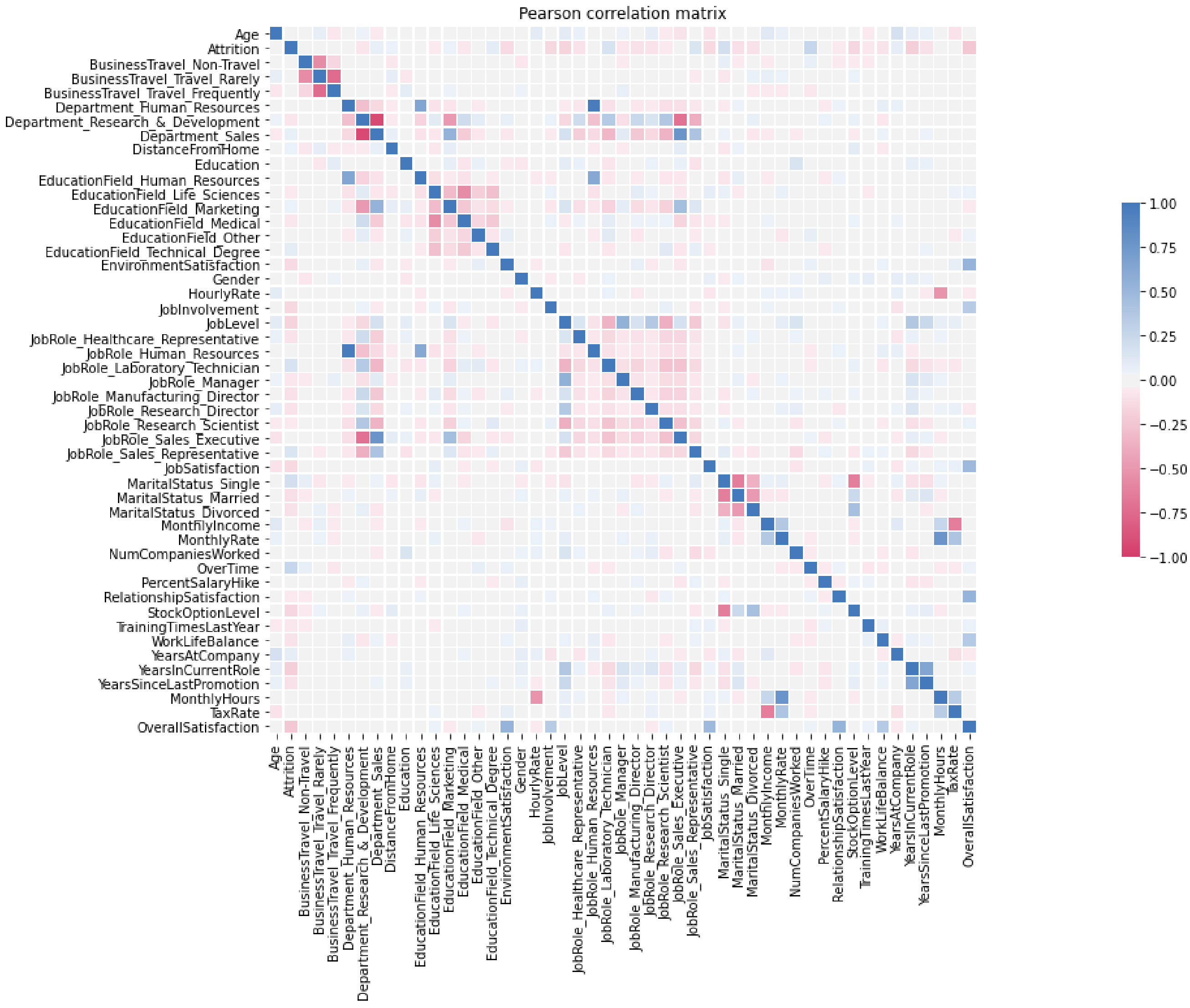


Figure 3: Pearson correlation matrix

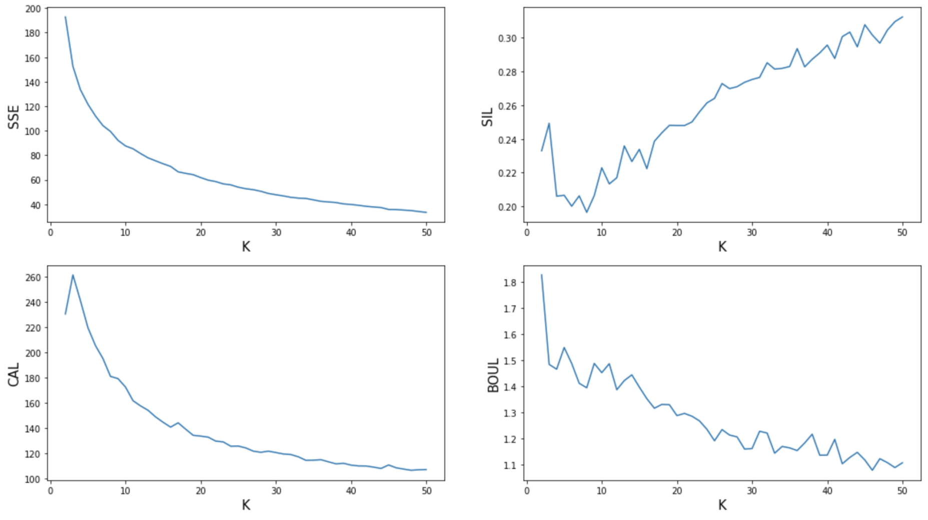


Figure 4: K-Mean's SSE, Silhouette, Calinski-Harabasz, Davies-Bouldin

2 Clustering

The first task, presented in this section, is to try to identify some clusters in the dataset by applying three different techniques: **center-based clustering** with K-means, **density-based clustering** with DBSCAN and

hierarchical clustering with the Agglomerative type. For this particular task, our data needs to be normalized, so remindful that clustering procedure are sensible to noise, we used **RobustScaler** as data normalizer.

2.1 K-Means

After considering different combinations of features, highlighted in Section 1.2, by narrowing or broadening at each trial the considered feature set, we got the best results using the **Euclidean distance** as metric and the following variables: **[JobLevel, StockOptionLevel, YearsAtCompany, YearsInCurrentRole, OverallSatisfaction]**.

We performed the algorithm considering the value of k from 2 to 50, and observing, for each single value of k, the metrics Sum of Squared Error, Silhouette, Calinski-Harabasz and Davies-Bouldin. We have chosen **3** as the **optimal value of k**, since, with respect to the measures mentioned above, we obtain the best compromise between the number of clusters, the sum of squared error and the quality measures that establish the separation between clusters in terms of more or less high similarity, as depicted in Figure 4.
We, however, investigate clusters having k initialized with values from 3 to 11, to understand how the centroids vary according to the increment of k. In Figure 5, for example, we can observe that increasing the value of k, increases the variance of the variables YearsAtCompany and YearsInCurrentRole within the clusters.

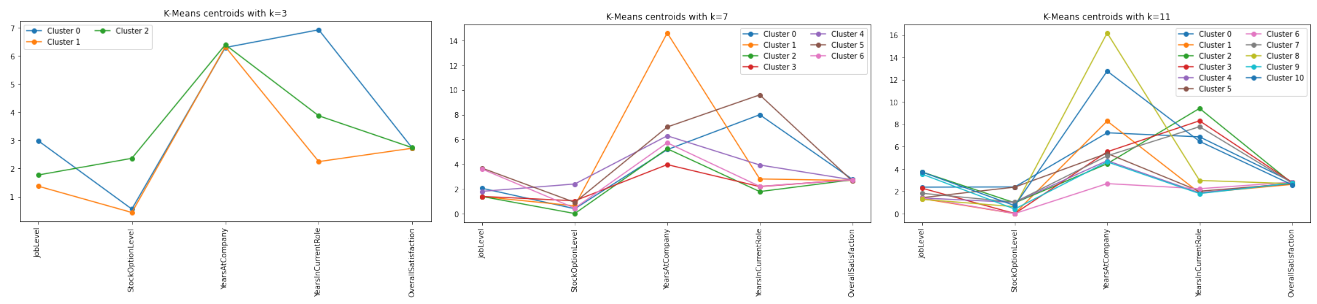


Figure 5: K-Means centroids for different k

We analyze the distribution of the variables within each of the three cluster found versus the distribution of the complete dataset. The **JobLevel** variable allows us to distinguish cluster 0 and cluster 1, as we observe that, with respect to the distribution in the complete dataset, a majority of lower values are captured by cluster 1, while cluster 0 collects higher values. The **StockOptionLevel** variable allows you to better define cluster 2, as it contains high and medium-high values, while clusters 0 and 1 contain only low and medium-low values. The **YearsInCurrentRole** variable allows to better characterize cluster 0, as this also contains the medium-high values of this feature, while the other two clusters mostly contain lower values. The distributions of the other two variables, **YearsAtCompany** and **OverallSatisfaction**, do not allow to establish a real difference between the clusters.

Twentythree variables demonstrate an intra-cluster distribution that reflects distribution within the original dataset. In conclusion we highlight how in **Cluster 1**, 115 out of the 153 leaving employees are clustered here, as shown in Figure 6 and Table 2, where K-Means' clusters state a **Jaccard coefficient** equal to 0.053, **0.226** and 0.060, respectively. From the contingency tables we can derive also Yes Attrition class label's **F-measure**, equal to 0.100, **0.368** and 0.113, respectively.

	Cluster 0	Others	Cluster 1	Others	Cluster 2	Others
Yes Attr	21	132	115	38	17	136
No Attr	244	486	356	374	130	600

Table 2: K-Means' clusters contigency tables

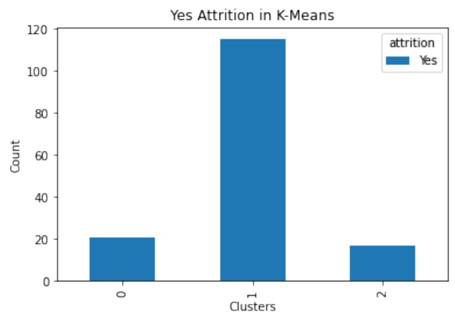


Figure 6: Yes Attrition in K-Means' clusters

2.2 DBSCAN

We decided to follow the same reasoning used for K-Means, thus we attempted to do the clustering over the same set of attributes, preferring again the **Euclidean distance**. We notice however that the previous tune to lower features' variation to better perform with K-Means, this time doesn't perform well with this clustering method. So we undergone multiple trials to narrow or to broaden the feature set, drawing from Section 1.2, whose final result is: **[Age, DistanceFromHome, MonthlyIncome, PercentSalaryHike, StockOptionLevel, TrainingTimesLastYear, YearsAtCompany, YearsInCurrentRole, MonthlyHours, OverTime, OverallSatisfaction, JobLevel, NumCompaniesWorked, Education]**. To choose the right Epsilon and Min-points, we adopted the Knee Method, by plotting the distance to the k-th nearest neighbour, where k was taken from [2, 10], due to the lower number of high dissimilarity among our points. The resulting curves, is shown in Figure 7, were we plot with the help of two Heat-maps DBSCAN's parameter tuning.

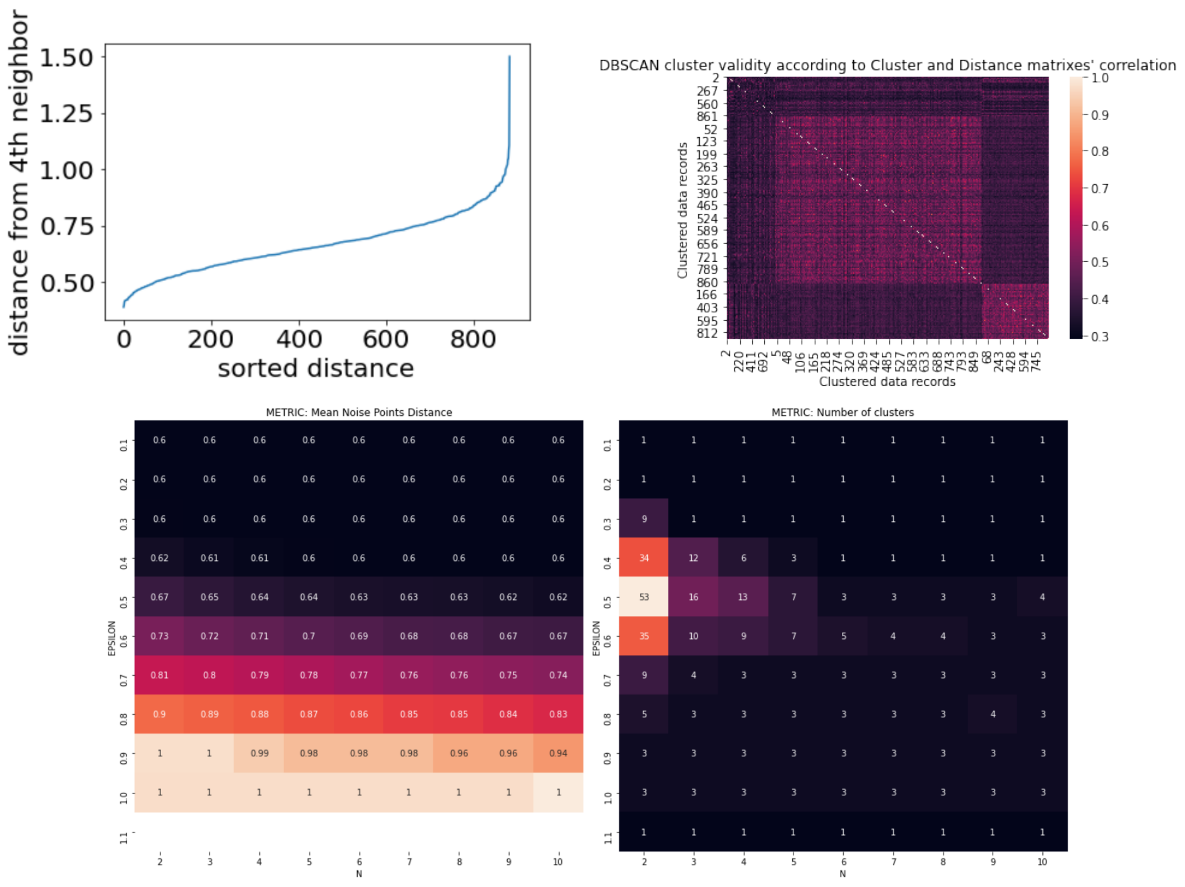


Figure 7: Knee Method. DBSCAN clusters validity via correlation matrix. Heatmaps displaying Mean Noise Points and Number of cluster for considerate neighbours N

Looking at the Heat-map we run multiple executions of DBSCAN, ranging form min=4 to min=8 and from esp=0.4 and esp=1.0. We were unfortunately unlucky, because we discovered than the clustering algorithm undergoes over-fitting above eps=0.8, under-fitting below eps=0.6 for any min. In the remaining range there is only space for a misleading classification of the dataset in three clusters. For example, for **min=4**, **esp=0.7** we have 3 clusters made out of 133 Noise Points, 567 points and 183 points, scoring one of the greatest **Silhouette coefficient score** and equal to **0.334**. The problem is that the we didn't detect **any significant changes in the three cluster's distribution**, meaning that the algorithm performed poorly in its clustering job. From clusters' correlation validity matrix in Figure 7 we can see how the noisy cluster (upper left) is clearly made out of outliers, while between the second cluster and the third cluster we detect only an employee's OvertTime specialisation in the third cluster (bottom right). The problem in our opinion is to be linked to our tight density dataset, which for DBSCAN is quite difficult to cluster well. For this reason we will discard DBSCAN as being the best approach to used with our data.

2.3 Hierarchical clustering

We performed the clustering on the same attributes used with K-Means and DBSCAN, trying also other combinations and using the features highlighted in Section 1.2. We found out that **K-Means' attributes set** is the one that better tunes with our trials. We decided to perform the hierarchical clustering with the **Euclidean distance**[8] as metric, and to perform, for each of those metrics ward, single, complete and average linkages, computing the full trees. Our founding are summarized bellow in Figure 8 and in Figure 9, where dendograms best cut was found by looking at Silhouette scores. The best out of the four is **ward-linkage**, since it **is the only able to detect meaningful clusters**, disregarding single-linkage highest Silhouette[9], as shown in Table 3. Our motivation resides in the fact that beside ward, all the other linkages split the data-set in extreme uneven cardinalities, due to the presence of noisy points which made them merged together in tightly blocks: the single-linkage suggests the presence of two cluster, the first made out of 882 points, the second composed by a single point; the complete-linkage finds two clusters, the first made of 791 points, the second by 92 points; the average-linkage discovers three clusters, one composed by 813 points, the second by 68 points and the third by 2 points. In the ward-linkage, the dendrogram suggests the presence of 3 main clusters, a larger one (468) and two smaller ones (275 and 468 points). Therefore ward-linkage's best cut confirms the best k evaluated for K-Means. Also clusters' cardinality and points presence among the two clustering methods are essentially identical to K-Means, as shown in Table 4. Ward-linkage clusters' **Jaccard scores** are 0.065, **0.213** and 0.065 and **F-measure** are 0.121, **0.351** and 0.123, respectively. Finally, notice that all dendograms merge at different distances, with the ward-linkage resulting in the highest distances, due to its different nature in creating clusters.

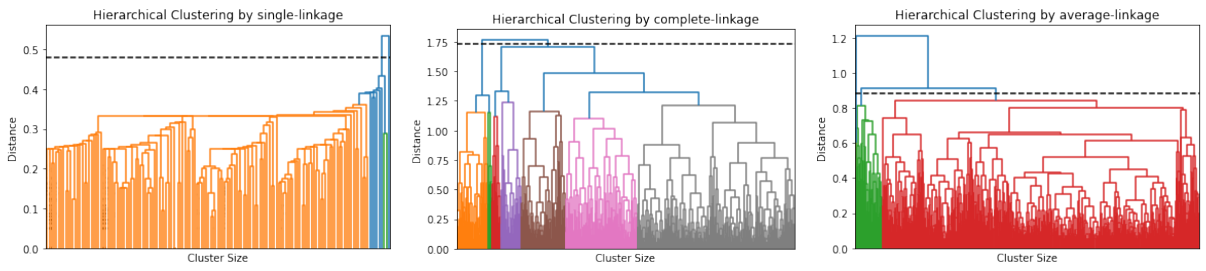


Figure 8: Hierarchical Clustering: single, complete and average linkage

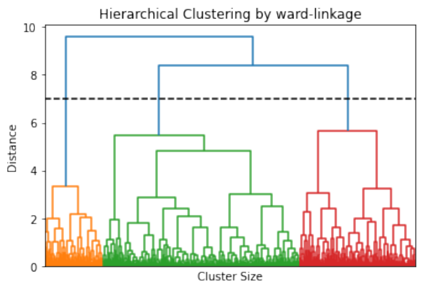


Figure 9: Hierarchical Clustering: ward linkage

Linkage:	Single	Comple	Average	Ward
SIL	0.339	0.179	0.163	0.225

Table 3: Linkages' Silhouette scores

	Cluster 0	Others	Cluster 1	Others	Cluster 2	Others
Yes Attr	26	127	109	44	18	135
No Attr	249	481	359	371	122	608

Table 4: Ward-linkage's clusters contigency tables

2.4 Final evaluation of the best clustering approach

With a poor performance of DBSCAN, K-Means results as the best option for this dataset, with a **Silhouette Score** of **0.249** over the **0.225** of the hierarchical clustering with the ward-linkage and in general a slightly higher value for the Jaccard Coefficient and F-measure. Please note that even though K-Means was the best solution amongst the ones explored, it is still an unsatisfactory result, which indicates that the **dataset is not suitable for clustering** or at least not for this basic techniques.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
MonthlyIncome	26997-16998	16985-10496	10482-9150	9129-4510	4502-4465	4444-4148	4127-4127	14107-1009
OverallSatisfaction	4.0-3.6	3.4-3.2	3.0-2.6	2.4-2.4	2.2-1.8	1.6-1.4	1.2-1.2	-
YearsInCurrentRole	16-6	5-5	4-3	2-2	1-1	0-0	-	-
YearsAtCompany	20-7	6-0	-	-	-	-	-	-

Table 5: DecisionTree discretisation values

[8]Our choice is justified both by the possibility to compare the hierarchical clustering with the previous two clusters methods (Euclidean distance was the preferred choice by both K-Means and DBSCAN) and by the fact that the ward-linkage accepted only Euclidean as affinity metric.

[9]DBSCAN and single-linkages silhouettes are quite similar, due to the fact that they work likewise.

3 Classification

In this section, we have applied two classification techniques in order to predict if a company's employee is intent to leave it or not. This kind of task requires a supervised classification model and we employed an **eager learner** (Decision Tree Classifier) and a **lazy learner** (KNeighborsClassifier). We noticed that the Training Set is highly unbalanced with respect to the target attribute object of the classification (Attrition):

- **Class=0** (No Attrition) : **82.7%**;
- **Class=1** (Yes Attrition): **17.3%**.

However, since in the first Data Mining course, we didn't faced up methods to cope with our unbalanced classification task, we won't address it. Moreover, to be able to test our predictive models we performed the same cleaning[10] described in Section 1.3 over the Test Set file, too, resulting in **219 records** and **24 features**. For this task our categorical data needs to be encoded as integer values (we evaluated both onehotencodig and sequential numerical encoding) and both for decision trees and KNN we evaluated plain, not discretized models and continuous variables discretized ones.

Parameter	Description	Values
criterion	function to measure the quality of a split	gini, entropy
max_features	# of features to consider for best split	None, auto, sqrt, log2
max_depth	max depth of the tree	range(1, n_features+1)
min_samples split	min # of samples required to split an internal node	range(10, 101, 10)
min_samples_leaf	min # of samples required to be at a leaf node	range(10, 101, 10)
class_weight	weights associated with classes	None, balanced

Table 6: Decision trees' hyper-parameters tuning

3.1 DecisionTreeClassifier

First, we divided the data set into **training** (**70%** of data) and **validation** (**30%** of data)[11] sets by using a **stratified splitting** that allowed us to preserve the percentage of samples for each class. Then, for the model selection, we performed a random grid search with **4-folds stratified cross-validation**[12] on the training set, to select model's best hyper-parameters, to validate on the validation set. The hyper-parameters optimization phase was carried out by combining the validation scheme, described above, and the range of parameters in the Table 6 for the random grid search. As models' goodness criterion we decided to use the value of ROC AUC because the accuracy alone is not a good evaluation option with class-imbalanced data sets. We trained the following models[13]:

1. onehot-encoded categorical variables without continuous variables binning (**Model 76**);
2. numerical-encoded categorical variables without continuous variables binning (**Model 17**);
3. onehot-encoded categorical variables with continuous variables DecisionTreeDiscretizer binning[14] (**Model 0**);
4. numerical-encoded categorical variables with continuous variables DecisionTreeDiscretizer binning (**Model 31**);

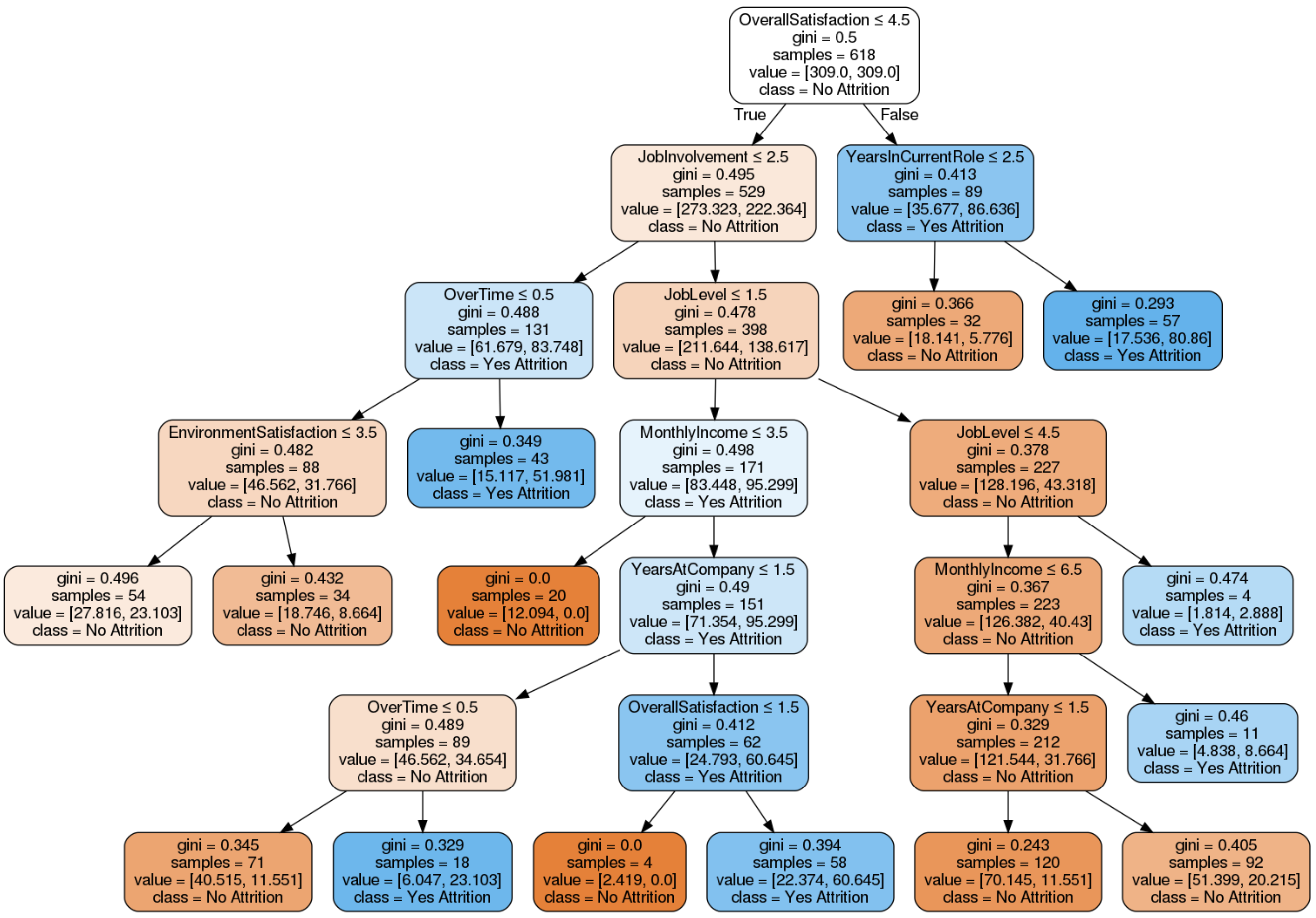
In particular, for models 17 and 31, we performed an ordinal encoding of their categorical variables [15], resulting in better positive predictive results, as we will soon see. Our validation process resulted in Model 17 highest ROC AUC (Figure 13, bottom, left). We decided, however, to run all models on test set, too, both for overfitting avoidance and potential performance improvement, so we proceed training all four models upon the full training set. Our models' best hyper-parameters tuning are displayed in Table 13, while models' feature importance is shown in Figure 11. We can notice that weighing categorical variables differently made **JobRole** emerged as a classification trait in both models, indeed, absent in one-hot encoded ones. All models eventually assigned great importance to **Overtime**, **JobLevel** and **OverallSatisfaction** features. From Figure 11 and Figure 12, we can notice that the one-hot encoded models tend to favour the remaining employees guessing (highest TN and lowest

[10]Test Set's imputation was made using Training Set's mode and self-imputer values.
[11]We considered a little less than 1/3 of our validation set in this way.
[12]In our trails, we found that 4 folds are the best trade-off among too many and to low minority class presence in each fold.
[13]Since Decision Trees are able to cope with correlated features, we kept our additional OverallSatisfaction features, too.
[14]For the decision tree discretisation, we followed the idea present here: https://towardsdatascience.com/discretisation-using-decision-trees-21910483fa4b, which discretizes a feature by classifying it using a simple decision tree and thanks to tree's splits, bins the feature accordingly.
[15]i.e. assigning higher, incremental, integers to hierarchical job positions or given higher priority to travel frequently employees.

FP rates among all models), with an increase in recall thanks to the continuous variables discretisation (Model 0). The same increase is observed in the numerical encoded models, too, leading in the discretized model greatest specialisation in leaving employees prediction (Model 31). The final **winner model**, however, is **Model 0**, since despite having a slightly lower test ROC AUC to Model 31, shows a greater accuracy, recall and f1-score. This boost in performance is to be linked to less number of features used in the classification process, thus leading to a lower model overfit. Model 0's decision tree is displayed in Figure 10.

Table 5 shows some useful discretized, continous variables. **Please note, that due to a semantic error in our trials, highest column's values are associate to lowest discretized encodings, and vice versa, against what the common sense may suggest**[16]. Keeping in mind this, we can see from our tree that leaving employees are classified accordingly by:

1. OverallSatisfaction ≤ 2.3[17] and YearsInCurrentRole ≤ 4;
2. OverallSatisfaction > 2.3, JobInvolvement ≤ 2 and OverTime $= 1$;
3. OverallSatisfaction > 2.3, JobInvolvement > 2, JobLevel ≤ 1, MonthlyIncome ≤ 9139, YearsAtCompany > 6, OverTime $= 1$;
4. OverallSatisfaction > 2.3, JobInvolvement > 2, JobLevel ≤ 1, MonthlyIncome ≤ 9139, YearsAtCompany ≤ 6, OverallSatisfaction ≤ 3.5;
5. OverallSatisfaction > 2.3, JobInvolvement > 2, JobLevel > 1, JobLevel > 4;
6. OverallSatisfaction > 2.3, JobInvolvement > 2, JobLevel > 1, JobLevel ≤ 4, MonthlyIncome ≤ 4137.



OverallSatisfaction ≤ 4.5 gini = 0.5 samples = 618 value = [309.0, 309.0] class = No Attrition
True
False
JobInvolvement ≤ 2.5 gini = 0.495 samples = 529 value = [273.323, 222.364] class = No Attrition
YearsInCurrentRole ≤ 2.5 gini = 0.413 samples = 89 value = [35.677, 86.636] class = Yes Attrition
OverTime ≤ 0.5 gini = 0.488 samples = 131 value = [61.679, 83.748] class = Yes Attrition
JobLevel ≤ 1.5 gini = 0.478 samples = 398 value = [211.644, 138.617] class = No Attrition
gini = 0.366 samples = 32 value = [18.141, 5.776] class = No Attrition
gini = 0.293 samples = 57 value = [17.536, 80.86] class = Yes Attrition
EnvironmentSatisfaction ≤ 3.5 gini = 0.482 samples = 88 value = [46.562, 31.766] class = No Attrition
gini = 0.349 samples = 43 value = [15.117, 51.981] class = Yes Attrition
MonthlyIncome ≤ 3.5 gini = 0.498 samples = 171 value = [83.448, 95.299] class = Yes Attrition
JobLevel ≤ 4.5 gini = 0.378 samples = 227 value = [128.196, 43.318] class = No Attrition
gini = 0.496 samples = 54 value = [27.816, 23.103] class = No Attrition
gini = 0.432 samples = 34 value = [18.746, 8.664] class = No Attrition
gini = 0.0 samples = 20 value = [12.094, 0.0] class = No Attrition
YearsAtCompany ≤ 1.5 gini = 0.49 samples = 151 value = [71.354, 95.299] class = Yes Attrition
MonthlyIncome ≤ 6.5 gini = 0.367 samples = 223 value = [126.382, 40.43] class = No Attrition
gini = 0.474 samples = 4 value = [1.814, 2.888] class = Yes Attrition
OverTime ≤ 0.5 gini = 0.489 samples = 89 value = [46.562, 34.654] class = No Attrition
OverallSatisfaction ≤ 1.5 gini = 0.412 samples = 62 value = [24.793, 60.645] class = Yes Attrition
YearsAtCompany ≤ 1.5 gini = 0.329 samples = 212 value = [121.544, 31.766] class = No Attrition
gini = 0.46 samples = 11 value = [4.838, 8.664] class = Yes Attrition
gini = 0.345 samples = 71 value = [40.515, 11.551] class = No Attrition
gini = 0.329 samples = 18 value = [6.047, 23.103] class = Yes Attrition
gini = 0.0 samples = 4 value = [2.419, 0.0] class = No Attrition
gini = 0.394 samples = 58 value = [22.374, 60.645] class = Yes Attrition
gini = 0.243 samples = 120 value = [70.145, 11.551] class = No Attrition
gini = 0.405 samples = 92 value = [51.399, 20.215] class = No Attrition

Figure 10: Best decision tree model (**Model 0**)

[16]We forgot to iterate in reverse order, when assign discretized, integer bins to our continous values. Model's performance, however, aren't influenced by this and since the high amount of our trials, we sticked with this counterintuitive encoding.

[17](2.4 + 2.2) / 2

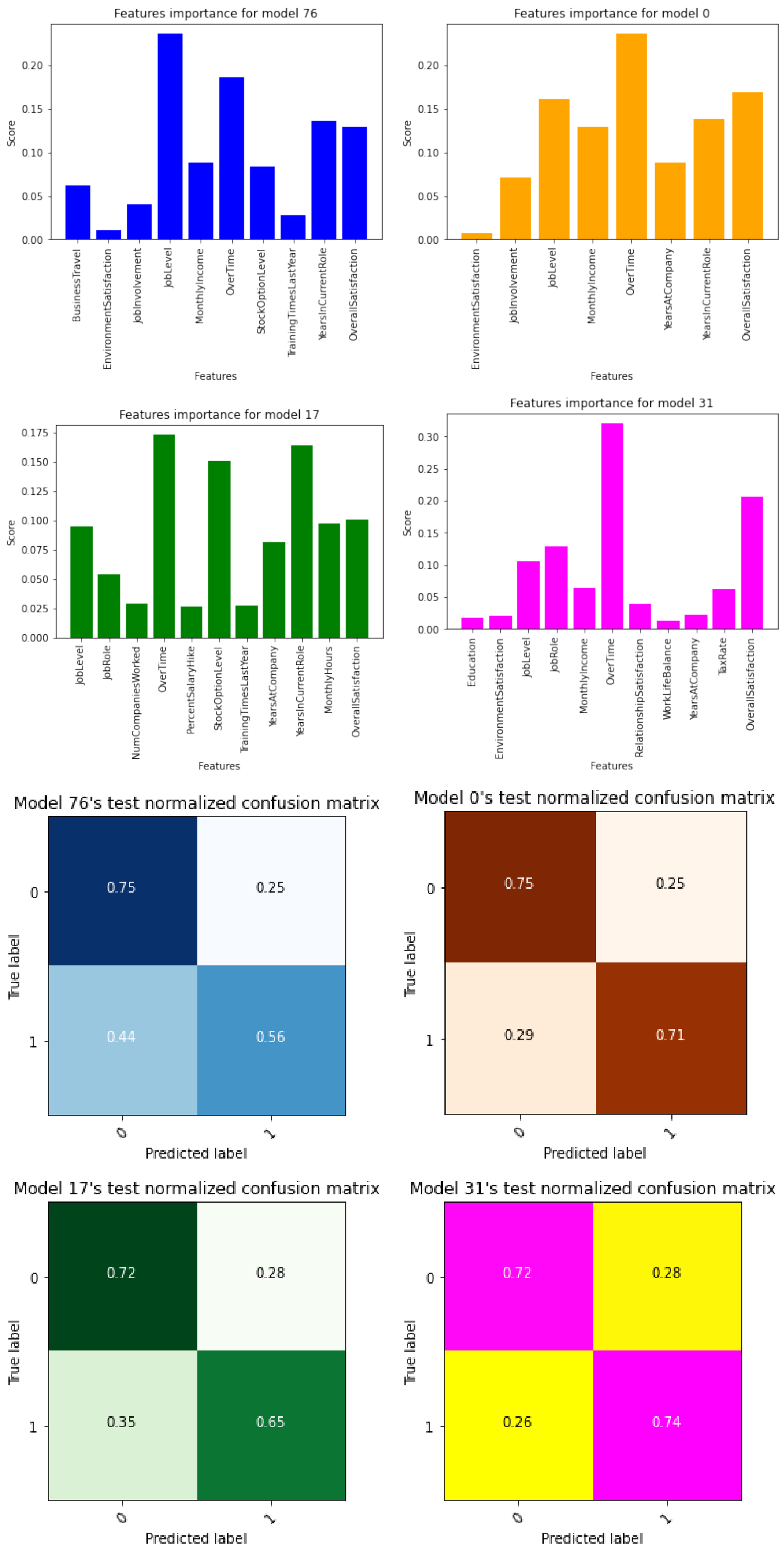


Figure 11: Decision Tree features importance and test contingency tables

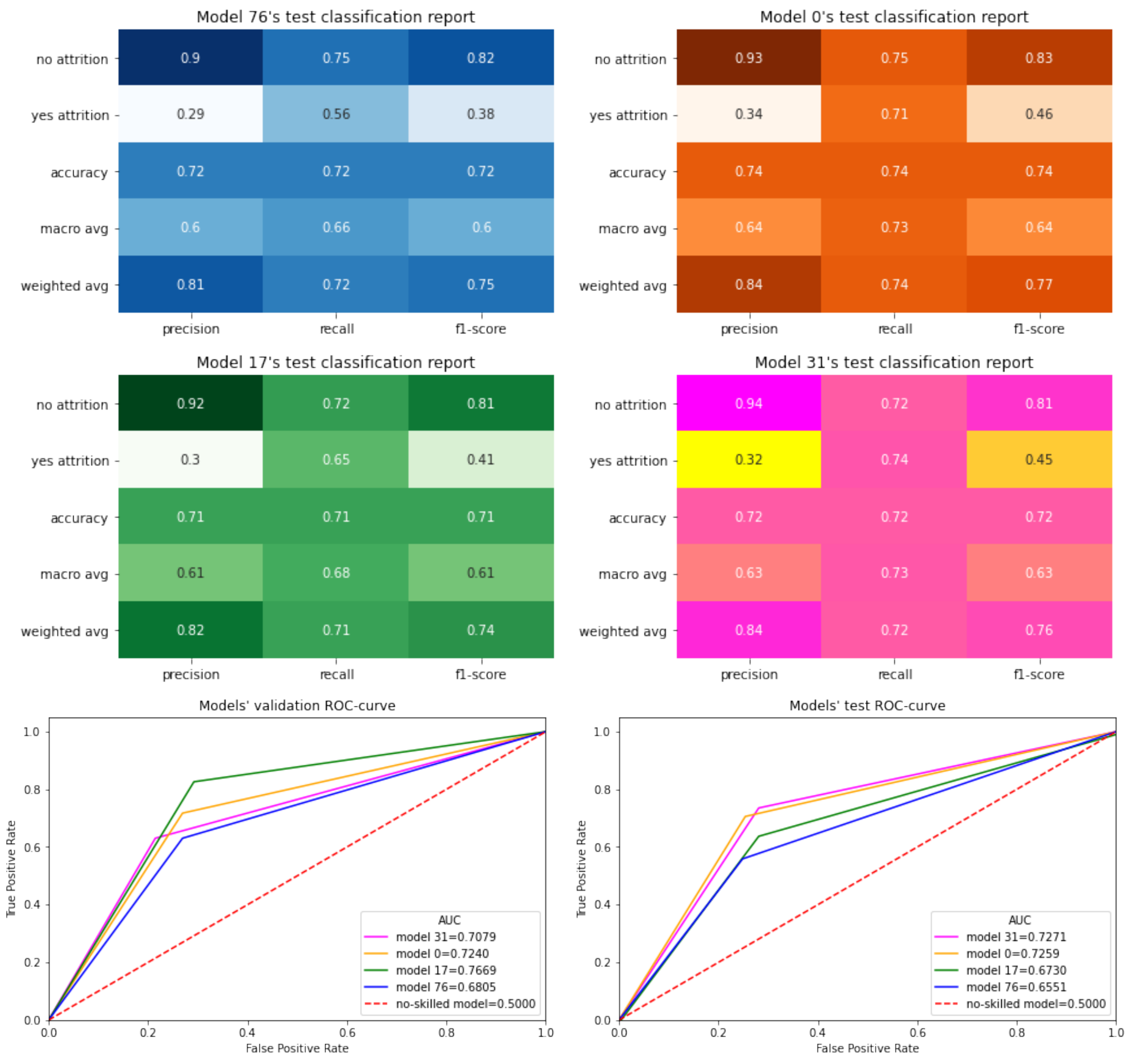


Figure 12: Decision Tree classification report and ROC AUC

	Model 76	Model 0	Model 17	Model 31
criterion	entropy	gini	entropy	gini
max features	auto	auto	log2	sqrt
max depth	5	6	11	15
min samples split	80	60	80	90
min samples leaf	30	2	5	10
class weight	balanced	balanced	balanced	balanced

Figure 13: Decision tree best hyper-parameters

3.2 KNeighborsClassifier

For the KNN task we considered four different versions of the dataset:

1. **36 Onehot**: the complete dataset with 36 features with onehot encoding;
2. **20 Onehot**: the dataset reduced from 36 features with onehot encoding, obtained eliminating the following variables: *YearsInCurrentRole, JobRole_ Manager_ Sales, JobRole_ Manager_ Human_ Resources, JobRole_ Sales_ Executive, JobRole_ Human_ Resources, YearsAtCompany, MonthlyIncome, MonthlyHours, Education, Gender, BusinessTravel_ NonTravel, PercentSalaryHike, BusinessTravel_ Travel_ Frequently, BusinessTravel_ Travel_ Rarely, TaxRate, JobRole_ Healthcare_ Representative*;
3. **24 Num**: the complete dataset with 24 features with numerical encoding;
4. **15 Num**: the dataset reduced by 24 features with numerical encoding, obtained eliminating the following variables: *YearsInCurrentRole, YearsAtCompany, MonthlyIncome, MonthlyHours, Education, Gender, BusinessTravel, PercentSalaryHike, TaxRate.*

The reduced datasets were obtained through a feature selection operation: since the KNN does not have its own method to evaluate the importance of the features, we used the *permutation_ importance* function on the training set. The *n_ repeats* parameter, set from us to 10, determines the number of times a feature is randomly shuffled, while we imposed to maximise the *roc_ auc* scoring. Performing the process several times and by taking returned features' runnings mean scores, made us able to discard the above mentioned features. Subsequently, since some continuous features were present in the dataset, we considered, for each analyzed dataset, three discretizers: **no discretization**, **discretization by quantiles** (with q = 4 bins, therefore by quartiles) and **discretization with K-Means** (with n_bins = 4). In the complete datasets all continuous variables have been discretized, in the reduced datasets only those remaining after the feature selection. Finally, since KNN tunes euclidean or manhattan distance through the *p* parameter, we normalized our data, using **MinMaxScaler**, **MaxAbsScaler**, **RobustScaler**, **StandardScaler**[18]. Please note that all this trials were performed to try to improve low classifier's scores, without, however, being able to cope to this drawback.

Parameter	Description	Values
n_neighbors	function to measure the quality of a split	range(2, 31)
weights	weight function used in prediction	uniform, distance
p	power parameter for the Minkowski metric	1, 2

Table 7: KNN's hyper-parameters

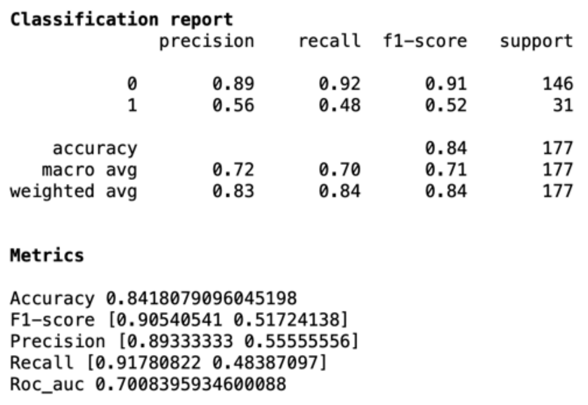
In the **Training phase**, for each dataset, we defined three possible, stratified splits (60% - 40%; 70% - 30%; 80% - 20%). On each training set we performed a **4-folds stratified cross-validation**, performing a grid search upon KNN's hyper-parameters showed in Table 7. In the **Validation phase**, the best hyper-parameters returned by the grid search were used to make the prediction on the stratified holdouts (validation set). As for the final **Test phase**, models, scoring the greatest ROC AUC scores upon the validation set, were used to classify the test set, training, the KNNClassifiers upon the entire training set.

In Table 8, Table 9 and Table 10, we highlighted for each data-set the highest scores on the validation set. If we observe the **Table 9** we discover that here we can find the best result among all. In fact, the best result in the Validation phase is that of the **20 Onehot data-set, 80%-20% split Qcut quantile discretization, MaxAbsScaler model**. This is the best classifier also in the Test phase, since the *roc_ auc_ score* registered for this is 0.65, the highest test score with respect to the others. Finally, KNN best models hyper-parameters are the following: **n_ neighbors=2, p=1, weights=distance**, while figures 14 and 15 report its prediction results in Validation and Test sets.

3.3 Discussion of the best prediction model

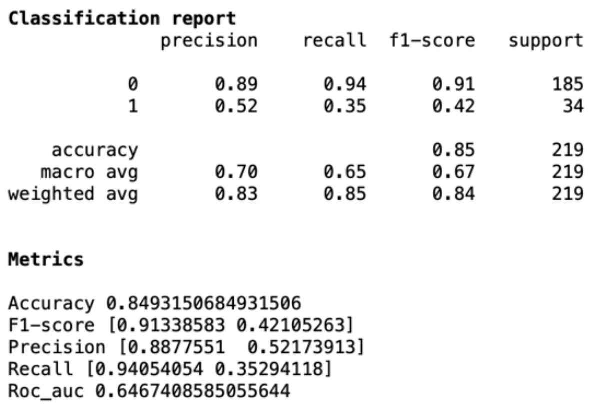
Comparing our best DecisionTreeClassifier (Model 0) with our best KNNClassifier (20 Onehot data-set, 80%-20% split, Qcut quantile discretization, MaxAbsScaler), with a ROC-curve AUC **0.7259** against a ROC-curve AUC **0.6467**, we state that the first makes better prediction than the second, both for the majority class and, more importantly, for the minority class.

[18]First we learnt the normalizer on given train folds and with it normalized the test fold. Afterwards we normalized holdout's training set and then applied it to the validation set. Then we normalized all our training set and applied the normalizer on the test set.



```
Classification report
              precision    recall  f1-score   support

           0       0.89      0.92      0.91       146
           1       0.56      0.48      0.52        31

    accuracy                           0.84       177
   macro avg       0.72      0.70      0.71       177
weighted avg       0.83      0.84      0.84       177

Metrics

Accuracy 0.8418079096045198
F1-score [0.90540541 0.51724138]
Precision [0.89333333 0.55555556]
Recall [0.91780822 0.48387097]
Roc_auc 0.7008395934600088
```

Figure 14: KNN best model's prediction results for Validation



```
Classification report
              precision    recall  f1-score   support

           0       0.89      0.94      0.91       185
           1       0.52      0.35      0.42        34

    accuracy                           0.85       219
   macro avg       0.70      0.65      0.67       219
weighted avg       0.83      0.85      0.84       219

Metrics

Accuracy 0.8493150684931506
F1-score [0.91338583 0.42105263]
Precision [0.8877551  0.52173913]
Recall [0.94054054 0.35294118]
Roc_auc 0.6467408585055644
```

Figure 15: KNN best models' prediction results for Test

No discretization																
Dataset	Validation												Test		
	MinMaxScaler			MaxAbsScaler			RobustScaler			StandardScaler					
36Onehot	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	51	10	0.57	53	8	0.56	51	10	0.57	50	11	0.57			
70-30	33	13	0.58	32	14	0.58	39	7	0.55	38	8	0.57	27	7	0.60
80-20	23	8	0.59	23	8	0.60	24	7	0.59	22	9	0.62			
20Onehot	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	45	16	0.61	39	22	0.62	47	14	0.60	43	18	0.63			
70-30	27	19	0.64	26	20	0.65	29	17	0.63	26	20	0.66	23	11	0.62
80-20	18	13	0.66	17	14	0.67	17	14	0.68	17	14	0.68			
24Num	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	47	14	0.56	44	17	0.58	51	10	0.56	52	9	0.55			
70-30	34	12	0.58	30	16	0.61	35	11	0.60	32	14	0.59	21	13	0.64
80-20	23	8	0.59	26	5	0.54	23	8	0.57	24	7	0.59			
15Num	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	47	14	0.59	45	16	0.61	49	12	0.59	39	22	0.61			
70-30	35	11	0.60	26	20	0.65	32	14	0.63	28	18	0.64	24	10	0.62
80-20	21	10	0.60	20	11	0.62	22	9	0.62	19	12	0.65			

Table 8: KNN No discretization validation and test classification scores

4 Association rules

In this section, we are performing association rules/pattern mining, whose goal is to find **frequent itemsets** and interesting **association rules** in the data-set. Before starting to explore all the possible frequent itemsets, it is necessary to apply some preliminary transformations. In fact, to implement the **apriori algorithm**, all continous features need to be transformed into categorical ones. The choice was to **discretize** each continous feature with **K-Means**, with **k=4**, since all intervals have almost the same length and obviously different within size.

4.1 Frequent, Closed and Maximal itemsets

To extract frequent, closed and maximal item-sets we used the apriori algorithm with different item-sets' sizes (**zmin** $\in [2, n\_features]$) and support thresholds (**supp** $\in$ **range(5, 101, 5)**). From Figure 16 (upper left) it is possible to notice that the number of item-sets decreases quite significantly in the first steps, from support threshold 0.1 to 0.2, highlighting how support threshold and item-sets' number are inversely proportional. Furthermore, as we expected there is no difference in considering frequent and closed item-sets, while maximal item-sets show a different behaviour until support threshold 0.2, due to the fact that their cardinality and within composition change less abruptly.

Since we are dealing with an unbalanced data-set, we further split our pattern mining discovery process, specialising, separately, Yes_Attrition item-sets. Our may reason in doing so is identified in Yes_Attrition

Qcut quantile discretization															
Dataset	Validation												Test		
	MinMaxScaler			MaxAbsScaler			RobustScaler			StandardScaler					
36Onehot	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	49	12	0.57	50	11	0.59	51	10	0.57	47	14	0.59			
70-30	32	14	0.58	37	9	0.57	39	7	0.56	37	9	0.58	26	8	0.60
80-20	24	7	0.59	25	6	0.57	21	10	0.60	21	10	0.63			
20Onehot	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	38	23	0.64	43	18	0.59	46	15	0.60	39	22	0.63			
70-30	33	13	0.62	33	13	0.62	37	9	0.59	28	18	0.64	22	12	0.65
80-20	17	14	0.69	16	15	0.70	20	11	0.65	17	14	0.69			
24Num	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	48	13	0.60	47	14	0.60	47	14	0.58	47	14	0.59			
70-30	32	14	0.58	34	12	0.56	32	14	0.59	33	13	0.58	24	10	0.63
80-20	23	8	0.57	25	6	0.58	25	6	0.58	23	8	0.60			
15Num	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	45	16	0.62	38	23	0.63	48	13	0.59	43	18	0.63			
70-30	31	15	0.64	32	14	0.63	38	8	0.57	27	19	0.65	24	10	0.62
80-20	16	15	0.67	16	15	0.67	21	10	0.64	18	13	0.66			

Table 9: KNN Qcut quantile discretization's validation and test classification scores

KBinsDiscretizer kmeans discretization															
Dataset	Validation												Test		
	MinMaxScaler			MaxAbsScaler			RobustScaler			StandardScaler					
36Onehot	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	54	7	0.55	50	11	0.58	45	16	0.61	50	11	0.58			
70-30	36	10	0.59	38	8	0.58	36	10	0.59	39	7	0.56	25	9	0.58
80-20	24	7	0.56	24	7	0.56	22	9	0.60	22	9	0.59			
20Onehot	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	47	14	0.59	42	19	0.60	46	15	0.58	39	22	0.62			
70-30	28	18	0.62	27	19	0.64	40	6	0.56	29	17	0.62	25	9	0.59
80-20	18	13	0.64	19	12	0.63	19	12	0.65	21	10	0.61			
24Num	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	51	10	0.56	50	11	0.58	50	11	0.57	48	13	0.60			
70-30	29	17	0.62	30	16	0.62	31	15	0.61	37	9	0.59	24	10	0.58
80-20	26	5	0.56	26	5	0.55	23	8	0.57	26	5	0.55			
15Num	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc	FN	TP	Roc
60-40	39	22	0.62	46	15	0.59	46	15	0.61	43	18	0.62			
70-30	32	14	0.63	28	18	0.63	36	10	0.60	28	18	0.64	23	11	0.63
80-20	18	13	0.64	20	11	0.62	22	9	0.63	24	7	0.59			

Table 10: KNN K-Means KBinsDiscretizer's validation and test classification scores

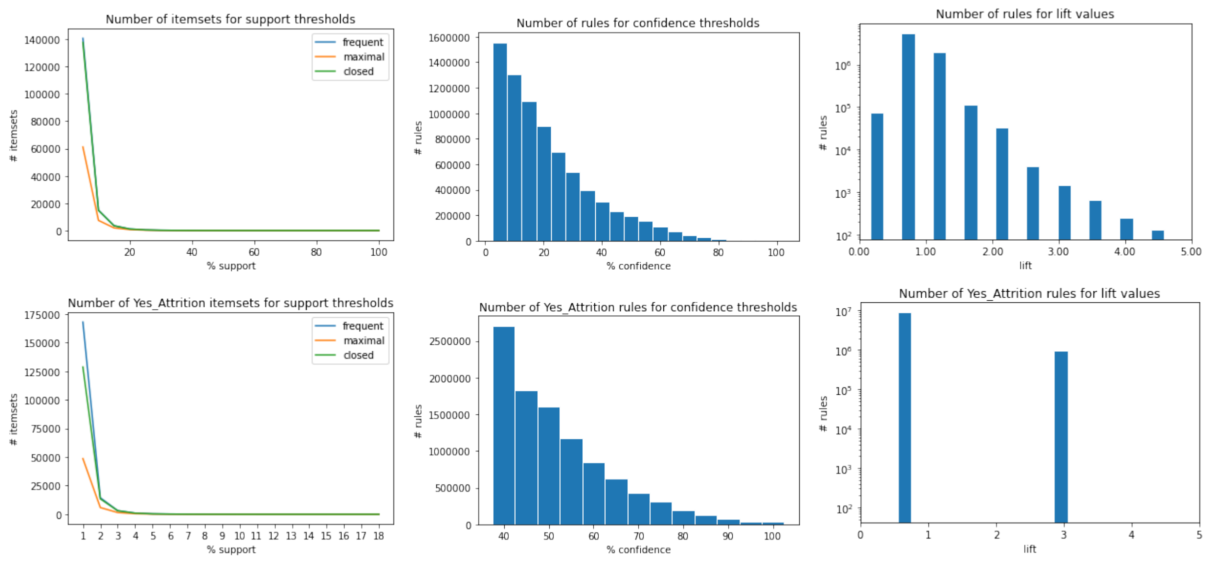


Figure 16: Item-sets' and Yes_Attrition item-sets' cardinality in support threshold changing (upper left and lower left), rules' cardinality in confidence threshold changing (middle upper and middle lower) and associated lift values (upper right and lower right).

item-sets' low support and as a consequence it requires a more sophisticated min_supp threshold tuning to avoid cutting out most or all Yes_Attrition patterns (we used **supp** $\in$ **[1, 18]**[19] to discover all Yes_Attrition item-sets). In Figure 16 (lower left), the same observations made previously apply, with an Yes_Attrition item-sets' numerical decrease between a support threshold of 0.02 and 0.03.
Table 11 summarizes some of our more interesting frequent item-sets:

1. most of frequent item-sets stating *No_ Attrition* present also *No_ OverTime* (in Table 11 we reported some examples where both are always present, to show how an increase in support leads to a decrease in item-sets' zmin, but this is not always the case);

2. 35% of our first level job roles are occupied by junior employees, having less than 4 years experience in given roles;

3. one third of employees is earning the lowest income but to it is applied the highest taxation (this is a remark we notice when studied variables' distribution, too, i.e. income and taxation are most most of the time inversely proportional as opposed to a more fair taxation system);

4. if a junior Laboratory Technician doesn't perform overtime, chances are that he stays. From this frequent item-set we learn thus that the main reason why junior Laboratory Technicians leaves has to be linked to heavy extra working ours and since they are the job role with the greatest turnover, this is an important remark to state;

5. 10% of employees are performing the greatest monthly working hours, without making actually overtimes. In addition, since these amount of working hours implies on calling 24 hours per day, ranging between 17 and 25 days a month[20], we derive that this strict working schedule is required for executive, managerial or professional positions.

Table 12, instead, displays some of our relevant Yes_Attrition frequent item-sets, among which all of them highlight how a low expertise (JobLevel=1), recent promoted positions (YearsInCurrentRole∈[0-4]) and a low number of previous companies experiences (NumCompaniesWorked∈[0-2]) make employees more flexible to leave. Doing overtime, don't holding any company's stock and working between 6 and 10 years at the company are other leaving employees' traits.

4.2 Associations rules

Due to the high decrease in the number of generated item-sets with a support threshold greater than 0.1 and 0.2, we decided to use **min_supp >= 0.1** to perform the extraction of **all association rules** and a support

[19]Yes_Attrition employees represent 17.3% of data-base's records.
[20]Stating again https://www.ibm.com/ibm/responsibility/policy11.shtml, employees are allowed at least one day off per seven-day week.

Support	Frequent itemsets
60-70%	{OverTime: No, Attrition: No}
50-60%	{WorkLifeBalance: 3, Attrition: No}
40-50%	{OverTime: No, BusinessTravel: Travel_Rarely, Attrition: No}
30-40%	{NumCompaniesWorked: [0-2], OverTime: No, Attrition: No}, {JobLevel: 1, YearsInCurrentRole: [0-4]}
20-30%	{YearsAtCompany: [0-5], MonthlyIncome: [1009-6992], OverTime: No, Attrition: No}, {TaxRate: [0.72-0.95], MonthlyIncome: [1009-6992]}
10-20%	{JobLevel: 1, YearsInCurrentRole: [0-4], MonthlyIncome: [1009-6992], OverTime: No, Attrition: No}, {JobRole: Laboratory Technician, JobLevel: 1, YearsInCurrentRole: [0,4], OverTime: No}
1-10%	{JobLevel: 1, NumCompaniesWorked: [0-2], Gender: Male, YearsInCurrentRole: [0-4], MonthlyIncome: [1009-6992], OverTime: No, BusinessTravel: Travel_Rarely, Attrition: No}, {MonthlyHours: [404-591], OverTime: No}

Table 11: Some frequent item-sets examples

Support	Frequent Yes_Attrition itemsets
10-15%	{YearsInCurrentRole: [0-4], Attrition: Yes}, {YearsInCurrentRole: [0-4], JobLevel: 1, Attrition: Yes}
5-10%	{YearsInCurrentRole: [0-4], JobLevel: 1, StockOptionLevel: 0, Attrition: Yes}, {YearsInCurrentRole: [0-4], JobLevel: 1, StockOptionLevel: 0, NumCompaniesWorked: [0-2], Attrition: Yes}
1-5%	{OverTime: Yes, YearsAtCompany: [6-10], JobLevel: 1, StockOptionLevel: 0, Gender: Male, NumCompaniesWorked: [0-2], YearsInCurrentRole: [0-4], BusinessTravel: Travel_Rarely, Attrition: Yes}, {OverTime: Yes, YearsAtCompany: [6-10], JobLevel: 1, MonthlyIncome: [1009-6992], Gender: Male, NumCompaniesWorked: [0-2], YearsInCurrentRole: [0-4], BusinessTravel: Travel_Rarely, Attrition: Yes}

Table 12: Some Yes_Attrition frequent item-sets examples

threshold of **min_supp >= 0.01** for **Yes_Attrition rules**. In the second case, despite Yes_Attrition item-sets generation's decrease happens above a support threshold of 0.02, we notice that some important Yes_Attrition are left out, so we prefer to be more conservative and include them in the association rule generation, too. The drawback of this choice is the heavy computation required to deal with the generation of any zmin, Yes_Attrition rule, having a support threshold greater or equal than 0.01. For this reason, we imposed a restriction uppon Yes_Attrition rules' zmin and confidence, searching them by using **zmin ∈ [4-9]**[21] and **conf ∈ range(40, 91, 10)**[22]. For the general association rule discovery, instead, we were less strict and generated any rule, having any zmin and confidence thresholds in **conf ∈ range(5, 101, 5)**. We selected both general and Yes_Attrition final rules, based on a tuned confidence threshold, since the more suitable lift values[23] were not particularly helpful in ARs classification (Section 4.3) and ARs imputation (Section 4.4)). The main reason why rules with an high lift didn't performed well with our tasks is to be linked to rules' low zmin. Thus, when, for example, classifying a record or performing a majority vote uppon multiple ARs to impute a missing value, using rules with high lift, leads to choose too general patterns and so performing poorly and with low accuracy. Instead, using strong rules wisely, leads to greater performance. For this purpose, after some tuning trials, we got **min_conf >= 0.40** for **all rules** and **min_conf >= 0.60** for **Yes_Attrition rules** as best confidence thresholds with which to select strong rules. Table 14 shows some rules examples, while Figure 16 displays some confidence and lift statistics. We can notice in particular that while rules number decreases with confidence increasing, lift is not as linear in its increase. Finally, in Table 13 we highlight the second rules, which despite having an height lift, is misleading since only 45 employees, matching rule's antecedents, are actually leaving, while 63 staying employees share the same antecedents and by our designed classifier, in Section 4.3, they will represent FP. The first, more specific strong

[21] In the Yes_Attrition item-sets generation phase, we noticed that the highest zmin is set to 9, while zmin ≤ 3 leads to Yes_Attrition item-sets with high support_count, but low accuracy in the eventual prediction of a leaving employee, since exist also No_Attrition item-sets having the same zmin and values, but an even greater numeracy in the data-set (this would lead, with great probability, to an high numper of FP).

[22] In our Yes_Attrition generation phase, we checked also if item-sets returned by the apriori algorithm were consistent with our training data-set's records. By doing this, we discovered that many item-sets, having a zmin ∈ [4-9], a support_count ∈ [9-13], matched all our corresponding leaving training records and none or only a handful (in any case strictly minor than Yes_Attrition records) of No_Attrition records, leading us to hypothesize the accurate recognition of some TP, using them. We choose 0.6 as a confidence base because in our trials this allowed us the most suitable equilibrium between TP and FP (please check Section 4.3, to get more insight in how we managed to classified records using ARs and and validate why this was the best choice).

[23] Lift takes into account the statistical dependence of the antecedent and the consequent and gives a better insight on how much information we actually gain by extracting a given rule, compared to the confidence.

Lift	High lift rules
4.0-5.0	{RelationshipSatisfaction: 1, Gender: Male} –>{OverallSatisfaction: [1.2-1.8]}
3.0-4.0	{JobLevel: 1, StockOptionLevel: 0, Gender: Male} –>{Attrition: Yes}
2.0-3.0	{TaxRate: [0.20-0.49], WorkLifeBalance: 3} –>{MonthlyIncome: [7094-13888]}
2.0-1.0	{OverallSatisfaction: 1, JobSatisfaction: 1, Attrition: No} –>Age: [31-38]}

Table 13: Some rules with high lift rules

Confidence	Strong rules
90-100%	{OverTime: Yes, YearsAtCompany: [6-10], JobLevel: 1, MonthlyIncome: [1009-6992], Gender: Male, NumCompaniesWorked: [0-2], YearsInCurrentRole: [0-4], BusinessTravel: Travel_Rarely} –>{Attrition: Yes}
90-80%	(TaxRate: [0.72-0.95], NumCompaniesWorked: [0-2], MonthlyIncome: [1009-6992], OverTime: No, BusinessTravel: Travel_Rarely} –>{Attrition: No}
70-80%	{JobLevel: 1, NumCompaniesWorked: [0-2], YearsInCurrentRole: [0-4], MonthlyIncome: [1009-6992], OverTime: No, Attrition: No} –>{WorkLifeBalance: 3}
60-70%	{Attrition: Yes, JobRole: Laboratory Technician, JobLevel: 1, StockOptionLevel: 0, Gender: Male, YearsInCurrentRole: [0-4], MonthlyIncome: [1009-6992], BusinessTravel: Travel_Rarely} –> {WorkLifeBalance: 1}

Table 14: Some strong rules examples

rule in Table 14, indeed, classifies correctly 9 leaving employees, with a FP = 0.

4.3 Classification using association rules

Since in this section we are interested in using pattern mining to get better knowledge in employees' attrition, in this third phase we disregard all rules that don't contain attrition's values as consequent. For the classification task using ARs, we re-adapted the dumb model (model which predicts always the majority class), deviating its predictions using Yes_Attrition strong rules. So, given a record which needs to be classified, if record's values matches at least one Yes_Attrition strong rules' antecenents, it is classified as belonging to class 1, belonging to class 0 if none Yes_Attrition strong rules' antecenents is exactly matched. Regarding Yes_Attrition strong rules selection, we first filtered them, keeping only the most specific ones, thus, preventing classifying too many records as leaving due too many strong rules, having a low zmin. These filtering phase, was undergone evaluating different confidence thresholds lower-bounds, and as we stating in Section 4.2, getting the best compromise between TP and FP and TN and FN prediction, using min_conf ≥ 0.60. Figure 17 shows the corresponding confidence threshold tuning and test classification's report and normalized confusion matrix. Given these best scores, we tried to enhance model's performance, by searching if among No_Attrition strong rules, existed more specif pattern than our Yes_Attrition rules, with which to balance the relative high number of FP. Unluckily, no previous No_Attrition strong rules was found sharing these characteristics and so no further improvements were undertake. Please notice, that the min_supp ≥ 0.40 for general strong rules (from which No_Attrition rules were derived) may have prevented these findings, but since general strong rules' confidence threshold tuning was done to be useful to Section 4.4 (were these other types of patterns are the leading actors), we felt bound to stick with them.

4.4 Imputation using association rules

Since in this section we are interested in using pattern mining to impute data-set's and test set's missing values, in this fourth phase we filtered all rules that have as consequent one of our NaN valued columns, described in Section 1.4. For the imputation task we used a copy of the original train and test data-sets, cleaned as stated in Section 1.3 and 3.3, without performing imputation. With our cleaned, imputed training data, in Section 4.2, we extracted a pool of rules, among which with the help of a confidence threshold, we draw out to substitute missing values based on a matching rule. Basically, we performed a reasoning similar as the one stated in Section 4.3: given a record having a missing entry[24], if record's other values match at least one strong rules' antecenents in our pool, it is imputed with rule's consequent. Of course, multiple matching can happen, so the final imputation is chosen through a majority vote among all matching rules' consequents. As for our strong rule pool, we filtered them, keeping only the most specific ones, thus, preventing imputing too many records with wrong values. Our best score in terms of imputation can be observed in Figure 18, obtained with a min_conf ≥ 0.40 (Table 15). We can notice that the absence of rules having many Age, MonthlyIncome, YearsAtCompany discretized values as consequents, prevent an accurate imputation process.

[24]Records having multiple missing entries are treated as having independent missing values, so iterated and imputed as many times as the number of their missing values.

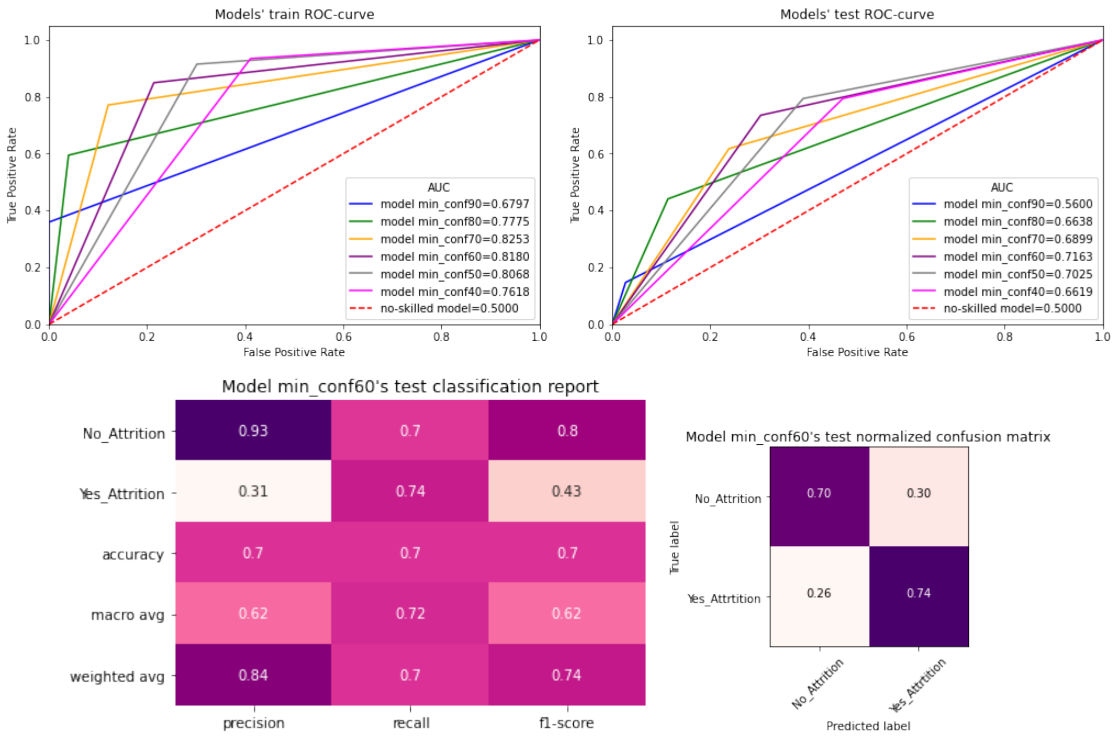
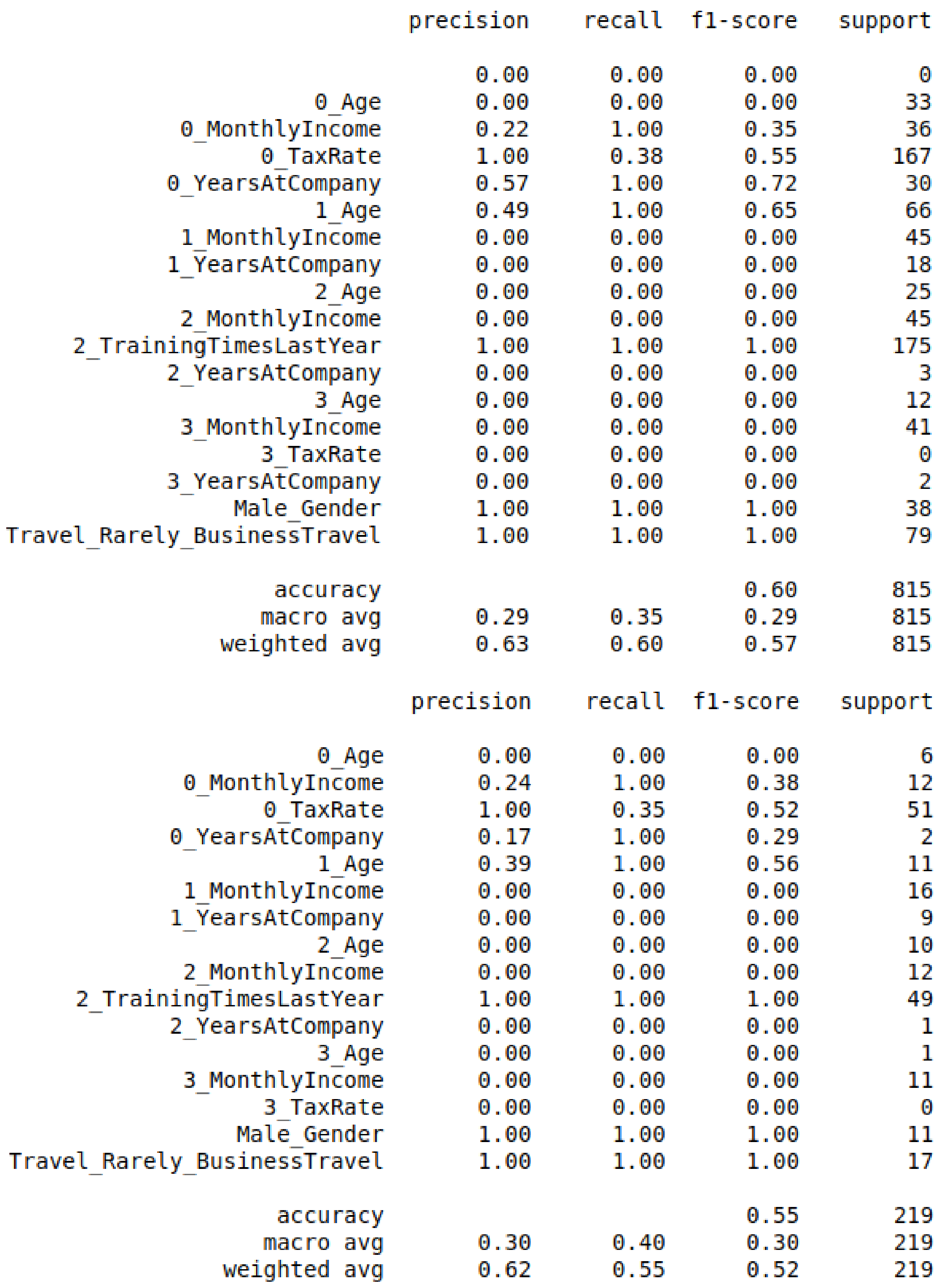


Figure 17: Association rules classification's ROC AUC train and test curves for confidence support tuning (upper), classification report and normalized confusion matrix (lower).

min_conf	**acc_train**	**acc_test**	**p_train**	**p_test**	**r_train**	**r_test**	**f_train**	**f_test**
>=60%	0.31	0.29	0.39	0.37	0.31	0.29	0.30	0.29
>=50%	0.51	0.46	0.43	0.39	0.51	0.46	0.46	0.40
>=40%	0.60	0.55	0.63	0.62	0.60	0.55	0.57	0.52
>= 30%	0.56	0.49	0.63	0.62	0.56	0.49	0.52	0.46

Table 15: Association rules imputation's confidence support tuning



```
                              precision    recall  f1-score   support

                                   0.00      0.00      0.00         0
                       0_Age       0.00      0.00      0.00        33
             0_MonthlyIncome       0.22      1.00      0.35        36
                   0_TaxRate       1.00      0.38      0.55       167
            0_YearsAtCompany       0.57      1.00      0.72        30
                       1_Age       0.49      1.00      0.65        66
             1_MonthlyIncome       0.00      0.00      0.00        45
            1_YearsAtCompany       0.00      0.00      0.00        18
                       2_Age       0.00      0.00      0.00        25
             2_MonthlyIncome       0.00      0.00      0.00        45
      2_TrainingTimesLastYear       1.00      1.00      1.00       175
            2_YearsAtCompany       0.00      0.00      0.00         3
                       3_Age       0.00      0.00      0.00        12
             3_MonthlyIncome       0.00      0.00      0.00        41
                   3_TaxRate       0.00      0.00      0.00         0
            3_YearsAtCompany       0.00      0.00      0.00         2
                 Male_Gender       1.00      1.00      1.00        38
Travel_Rarely_BusinessTravel       1.00      1.00      1.00        79

                    accuracy                           0.60       815
                   macro avg       0.29      0.35      0.29       815
                weighted avg       0.63      0.60      0.57       815
```

```
                              precision    recall  f1-score   support

                       0_Age       0.00      0.00      0.00         6
             0_MonthlyIncome       0.24      1.00      0.38        12
                   0_TaxRate       1.00      0.35      0.52        51
            0_YearsAtCompany       0.17      1.00      0.29         2
                       1_Age       0.39      1.00      0.56        11
             1_MonthlyIncome       0.00      0.00      0.00        16
            1_YearsAtCompany       0.00      0.00      0.00         9
                       2_Age       0.00      0.00      0.00        10
             2_MonthlyIncome       0.00      0.00      0.00        12
      2_TrainingTimesLastYear       1.00      1.00      1.00        49
            2_YearsAtCompany       0.00      0.00      0.00         1
                       3_Age       0.00      0.00      0.00         1
             3_MonthlyIncome       0.00      0.00      0.00        11
                   3_TaxRate       0.00      0.00      0.00         0
                 Male_Gender       1.00      1.00      1.00        11
Travel_Rarely_BusinessTravel       1.00      1.00      1.00        17

                    accuracy                           0.55       219
                   macro avg       0.30      0.40      0.30       219
                weighted avg       0.62      0.55      0.52       219
```

Figure 18: Association rules imputation on train (upper) and test data(bottom).