UNIVERSITY OF PISA
DEPARTMENT OF INFORMATICS

ASSIGNMENT OF FINAL PROJECT
DATA MINING COURSE

Dataset
Human Resources Analytics

Akeksandra Krzywańska, Tri Hong Nguyen,
Michal Punčochář, Jasmin Seyfi

Teacher
Prof. Dino Pedreschi,
Prof. Anna Monreale,
Dr. Riccardo Guidotti.

January 4, 2018

Table of Contents

Abstract. The assignment is about the project in Data Mining (DM) course at University of Pisa. In this assignment, four tasks of DM including: Data understanding, Clustering, Association Rules and Classification are implemented based on the data set Human Resources Analytics (HRA) [1]. In more detail, the data understanding task is to generally get the information from the data set, while the others try to specifically archive information such as: models or rules for prediction of new data.

1 Introduction

The DM part in Computer Science nowadays have become a hot trend, because of the increasingly large number of various data. With the plenty of data everyday, there are questions about understanding or utilization these data for different tasks.

In DM topic, there are some technologies such as: Data Understanding (DU), Clustering, Association Rules (AR) and Classification. From the diversity of tasks, these technologies can be utilized to analysis and understand the data. In addition, the assignment exploits the data set (HRA) [1] based on these technologies. Particularly, in the data set, there are 10 attributes such as: satisfaction_level, last_evaluation, number_project, average_montly_hours, time_spend_company, Work_accident, left, promotion_last_5years, sales and salary. From these attributes, some techniques in DU are used to exploit the data set to clearly gain common information. Moreover, in clustering part, techniques including: K-means, DBScan and Hiearchical are used for clustering similar records or employees to bigger groups or clusters. Meanwhile, the AR task is about rules extraction and frequent item sets. This task tries to gain rules which are general or common. These rules can be used for other tasks such as: filling missing values in the data set. Finally, to classify values in a attribute, the classification can be apply to build a model which can predict the values of the attribute from a new data. In the classification task, the decision tree and random forest techniques are utilized, because of the efficiency with the small data set.

The remaining of the assignment are represented: Section 2 describes about the Data Understanding. The Clustering is showed in Section 3, while the Association Rules and Classification are illustrated in Sections 4 and 5, respectively.

2 Data Understanding

2.1 Data sematics

The analised data set consists of 14999 records with 10 different variables. Three of those attributes *work_accident, promotion_last_5years* and *left* are binary attributes. The variable salary with the values *low, medium* and *high* are ordinal while the variable *sales*, which describes the workfield, is nominal. The rest of the attributes is numerical of which two are continuous in the Interval between 0 and 1. These two features are *satisfaction_level* and *last_evaluation*.

2.2 Distribution of the variables and statistics

By looking at the statistics that belong to the features that are depicted in table 1 we can conclude that none of the binary attributes follow an equal distribution. All of them are biased toward 0 or the negation. This is shown in Figure 1.

Only the attribute *number_project* can be considered normally distributed, *time_comapany* can be viewed as geometric distributed and the the variables *average_monthly_hours, last_evaluation* and *satisfaction_level* are more or less multimodal distributed.

These different distributions should be kept in mind when analysing the results later.

In regards to the statistics of each of the attributes one should look at the numerical features. They contain more valuable information than the binary ones.

A look at table 1 reveals the fact that the standard deviation for the attribute is the greatest for *satisfaction_level* with about 40% of the mean value. The feature *last_evaluation* has smallest standard deviation followed by *average_monthly_hours* with, in both cases,less than 25% of their mean value.

Table 1: Statistics of all attributes

	satisfaction_level	last_evaluation	number_project	montly_hours	time_in_company	accident	left	promotion
count	14999	14999	14999	14999	14999	14999	14999	14999
mean	0.612	0.716	3.803	201.050	3.498	0.144	0.238	0.021
std	0.248	0.171	1.232	49.943	1.460	0.351	0.425	0.144
min	0.090	0.360	2.000	96.000	2.000	0.000	0.000	0.000
25%	0.440	0.560	3.000	156.000	3.000	0.000	0.000	0.000
50%	0.640	0.720	4.000	200.000	3.000	0.000	0.000	0.000
75%	0.820	0.870	5.000	245.000	4.000	0.000	0.000	0.000
max	1.000	1.000	7.000	310.000	10.000	1.000	1.000	1.000

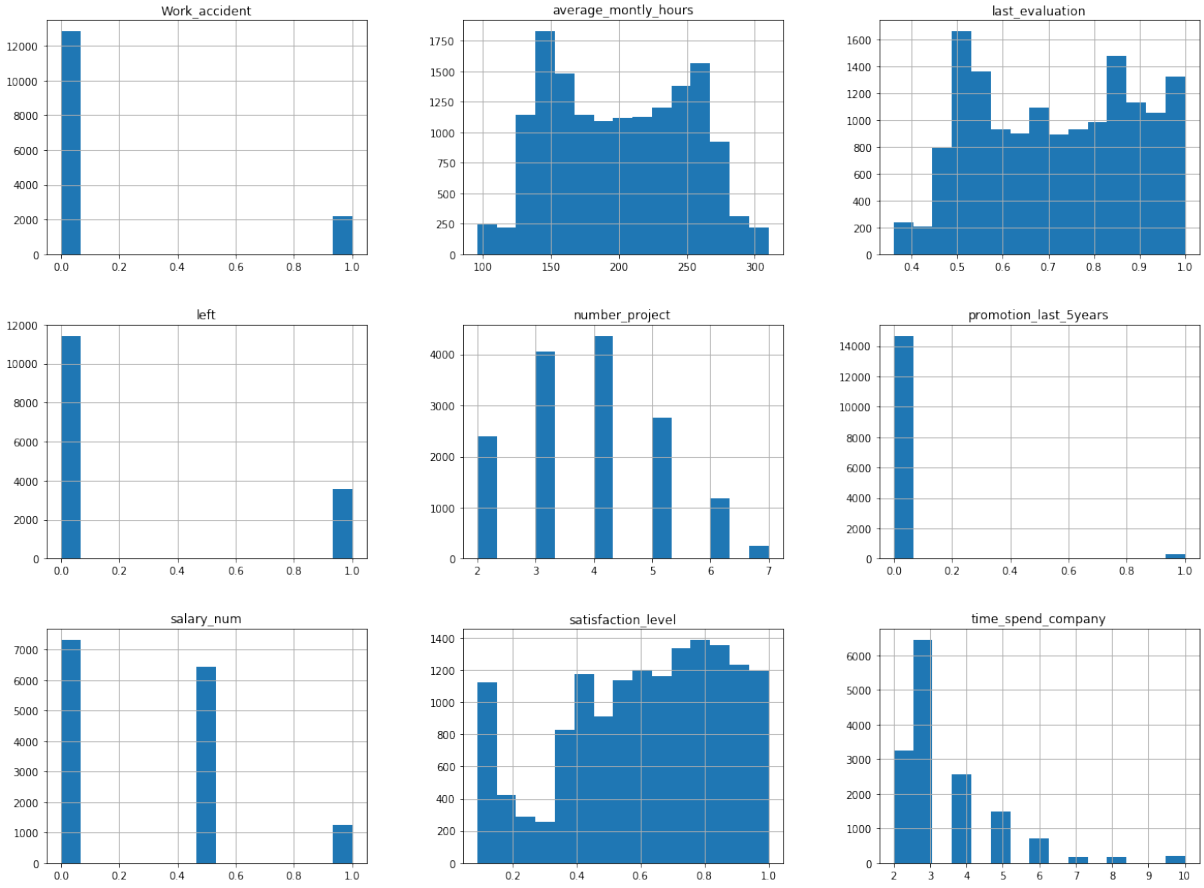


Fig. 1: Histogram of all the attributes. To computed the bin's size we used Sturge's law

2.3 Assessing data quality

When we looked at the data we did not find any missing values. On that account we did not fill in any values or exempt the concerned records from our analysis. Thus all instances were used in the following analysis.
In regards to outliers we could not identify any.
It can be said, that there is a strong bias in the data set, that we have to treat cautiously when coming to conclusions. For one there is the bias in the binary features to consider. But there is also an unbalance in the representation of the different workfields in the company. Most of the employees working in the company are working in sales, support, and technical sectors. Together they make up more than half of the amount of the employees, but the respective percentage of people who left the company is greater in the other workfields. This instance is illustrated in Figure 2.

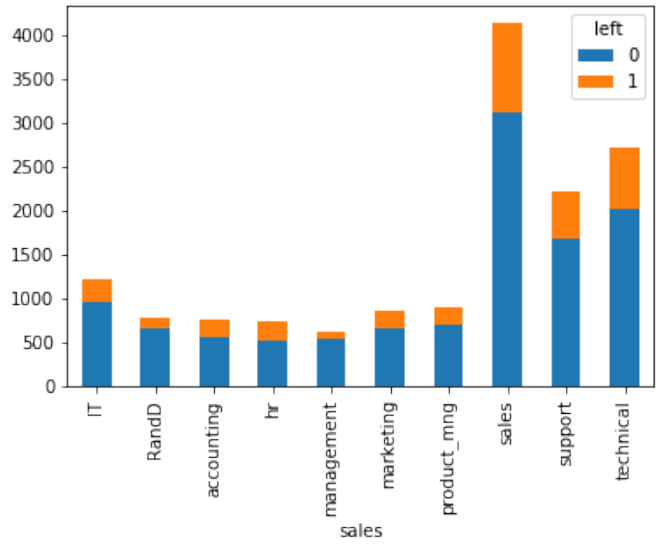


Fig. 2: Histogram of the attribute *sales* in regard to the attribute *left*

2.4 Variable transformation

We transformed the categorical variable salary into a numerical attribute by assigning values to the different categorical values. To the ordinal value *'low'* we assigned the value 0, we replaced *'medium'* with 0.5 and *'high'* was substituted with 1. By doing so we obtained nine attributes with numerical values. Afterwards we computed the correlation matrix with those nine variables.

Table 2: Correlation matrix of the features

	satisfaction_level	last_evaluation	number_project	montly_hours	time_in_company	accident	left	promotion	salary_num
satisfaction_level	1.000	0.105	-0.143	-0.020	-0.101	0.059	-0.388	0.026	0.050
last_evaluation	0.105	1.000	0.349	0.340	0.132	-0.007	0.007	-0.009	-0.013
number_project	-0.143	0.349	1.000	0.417	0.197	-0.005	0.024	-0.006	-0.002
montly_hours	-0.020	0.339	0.417	1.000	0.128	-0.010	0.071	-0.004	-0.002
time_in_company	-0.101	0.132	0.197	0.128	1.000	0.002	0.145	0.067	0.049
accident	0.059	-0.007	-0.005	-0.010	0.002	1.000	-0.155	0.039	0.009
left	-0.388	0.007	0.024	0.071	0.145	-0.155	1.000	-0.062	-0.158
promotion	0.026	-0.008	-0.006	-0.004	0.067	0.039	-0.062	1.000	0.098
salary_num	0.050	-0.013	-0.002	-0.002	0.049	0.009	-0.158	0.098	1.000

2.5 Pairwise correlation and elimination of redundant variable

Using the correlation matrix we computed one can see, that the highest correlation can be found between the variables *average_monthly_hours* and *number_of_ projects* with a value of 0.417.This indicates, that the more projects a person is working on, the more time he spends at work. The second highest value is detected in the negative correlation between the attributes *satisfaction_level* and *left*. The higher the satisfaction level, the less likely the people will leave the company. These correlations are as we expected. There was not one correlation that suprised us, all were to our prediction.
The smallest correlation value can be found between *salary_num* and *number_of_projects*. The value for this pair is -0.002. Because it is almost 0, it can be said, that there is no correlation between these two variables, they are independent from each other.
In 32 cases of the 81 computed correlations the resulting value is indicating a negative correlation between two attributes of the matrix.

Since none of the values depicted in table 2 are higher than 0.5 they are not redundant and therefore cannot be eliminated.

3 Clustering analysis

3.1 K-means clustering

Choice of attributes and distance function, preprocessing First we have dropped all non-numerical attributes. That left us with *satisfaction_level, last_evaluation, number_project, average_montly_hours, time_spend_company.* Then we have normalized all attributes into a $[0, 1]$ range.
And as a distance function we chose standard Euclidean distance.

Choice of best K We have run the K-means algorithm for $K = 2, 3, \ldots, 20$. For each K, the algorithm was run with 10 different initializations and the best result in terms of SSD was chosen. This was the default behaviour of scikit-learn implementation.
We have plotted Sum of squared distances (SSD), relative loss of SSD and a Silhouette coefficient (see the figure 3). Relative loss of SSD is defined as follows:

$$\text{Relativeloss}_K = \frac{\text{SSD}_K - \text{SSD}_{K+1}}{\text{SSD}_{K+1}}.$$

The silhouette coefficients were computed on a random sample of size 10000 for computational reasons.
The SSD plot doesn't tell us much, the curve turns very smoothly. The relative loss shows that the first two points corresponding to losses from $K = 2$ to $K = 3$ and from $K = 3$ to $K = 4$ are relatively high compare to others. Based on this we chose $K = 4$, because the further transition from $K = 4$ to $K = 5$ doesn't provide us with great relative improvement of SSD. Also, higher number of clusters would be more difficult to interpret.
Silhouette coefficients for $K \geq 3$ are very similar in a range $0.25 - 0.28$.

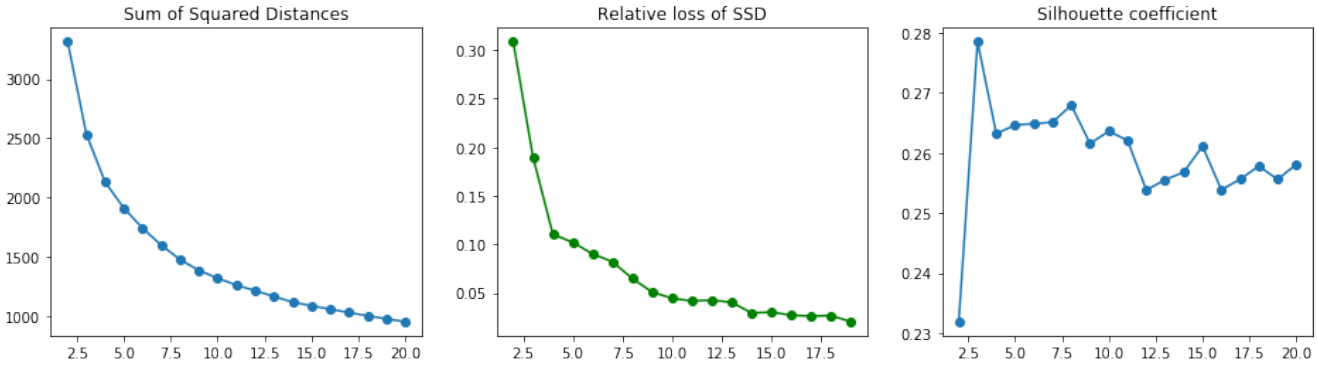
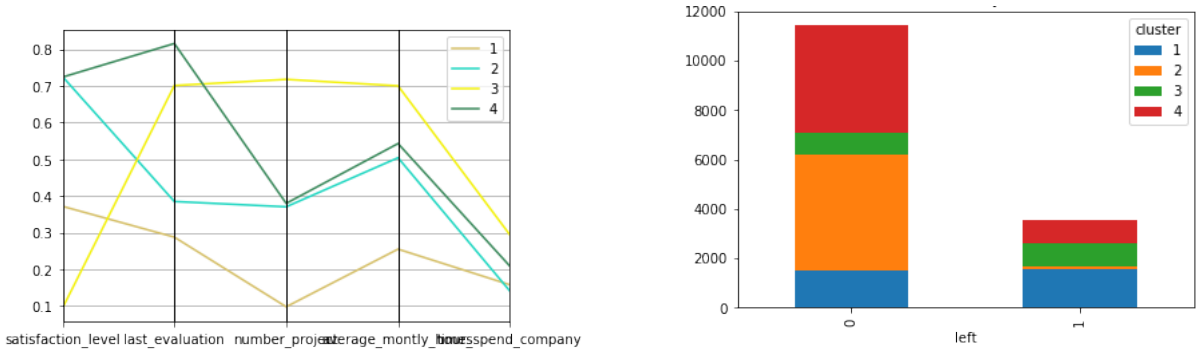


Fig. 3: Plots of SSD, relative loss of SSD and Silhouette score as a function of K.

Analysis of clusters and centroids The final clustering has $SSD = 2129.33$ and silhouette coefficient 0.267. See the Table 3 for the centroids coordinates.
We have plotted parallel coordinates of centroids (see the Figure 4a) and distribution of *left* class in clusters (see the Figure 4b).
Looking at the figures, there are many things that can be said about this clustering. For example, we can see that the least satisfied workers are in cluster 3. At the same time they also have the highest number of projects and the highest average time spent in company. Around half of them left the company.
People in cluster 1 are not very satisfied either. Although they didn't have many projects assigned and spend the least time at work. They were evaluated the most recently. Around half of them, too, left the company.
The clusters 2 and 4 appear very similar. The difference is that people from cluster 2 were evaluated more recently and also almost no one left the company from cluster 2 while some people from cluster 4 did leave.

Table 3: Cluster's centroids. Values are denormalized into their original scale.

Cluster	*satisfaction_level*	*last_evaluation*	*number project*	*average_monthly_hours*	*time_spend_company*
1	0.43	0.54	2.49	150.59	3.26
2	0.75	0.61	3.85	203.89	3.13
3	0.18	0.81	5.59	245.83	4.35
4	0.75	0.88	3.90	212.17	3.67



(a) (b)

Fig. 4: (a): Parallel coordinates of centroids. Values are normalized, (b): Cluster label rate by *left* classes

3.2 DB Scan clustering

Choice of data attributes and algorithm parameters For DBScan we have used the same attributes and distance function as for k-means. We have also rescaled all features into $[0, 1]$ range.
Here we had to choose two parameters: k and ε. Then, for some point, minimum of k points (including itself) have to be closer than ε for the point to be set as a core point.
DBScan algorithm, at least as implemented in scikit-learn, is more intensive on memory. Therefore we have randomly downsampled the dataset to 5000 samples and run the algorithm on the sample. The rest of the points were assigned to clusters with k-NN classifier, where k was the same as the k used for DBScan clustering. Noise points were treated as one more cluster, which is not very sound, because then a point gets classified as noise only when it is surrounded by other noise points. But the results in the end show that the silhouette score didn't get worse compared to the sample, so we have stayed with this simplified approach.
Note that we always measured Silhouette score only on the non-noise points.
First, we have plotted distances to k-th nearest neighbour from all points for different values of k, see the Figure 5. The curves looked very similar, we liked the most $k = 10$. That suggested to pick ε in a range around $0.2 - 0.3$. Then we have run the algorithm for different values of ε. The clustering were not very satisfactory, see the Table 4. Lower values of ε provided us with too many clusters while higher values put vast majority of points into a single cluster. We have chosen somewhat a middle way: $\varepsilon = 0.19$, which gave us 10 clusters, 15.9% of noise points and silhouette score 0.142.
We have fitted a 10-NN classifier on the sample and predicted the cluster label for the rest of the points. Then we merged these two groups. We have obtained a clustering of the full dataset with silhouette score 0.146 and 14.7% of noise points. See the Table 5 for a distribution of examples in clusters.
We can see that in some clusters there are only few examples. We have decided to treat them as noise as well. Therefore we selected the clusters with less than 200 examples and set them all to noise. We ended up with 6 clusters, silhouette score 0.158 and 16.9% of noise points.

Analysis of obtained clustering We have plotted distribution of class *left, satisfaction_level* and *time_spend_company* in the cluster, see Figures 6a and 6c. On the first sight all looks interesting, but when we plotted the distribution of *number_project* (Figure 6d), we realized that the DBScan clustered all examples basically just by this attribute. Therefore the whole clustering turned out being quite useless.
The solution to this issue could be to remove also all discrete attributes along with the non-numerical from the clustering, but then we would cluster by just three attributes which is not very meaningful nor informative.

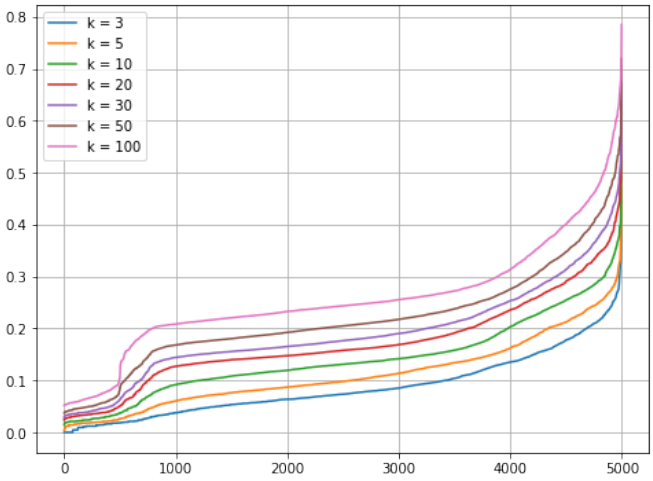


Fig. 5: Distance from all points to their k-th nearest neighbour, sorted by distance

Table 4: Clustering obtained for different choice of ε

ε	# of clusters	Silhouette score	Number of examples in clusters (-1 is noise)
0.10	47	0.324402	{0: 511, -1: 3055, 43: 10, 12: 77, 19: 65, 1: ...
0.11	39	0.102353	{0: 514, 6: 12, -1: 2589, 3: 472, 21: 21, 4: 3...
0.12	31	0.047462	{0: 516, 1: 548, -1: 2180, 5: 165, 2: 554, 3: ...
0.13	11	0.107176	{0: 526, 1: 971, 4: 137, -1: 1790, 2: 1084, 3:...
0.14	12	0.067739	{0: 537, 1: 1084, 4: 447, 2: 1228, -1: 1359, 3...
0.15	8	0.123547	{0: 552, 1: 1125, 2: 604, 3: 1263, -1: 1163, 4...
0.16	9	0.162462	{0: 566, 1: 1152, 2: 647, 3: 1277, 6: 14, -1: ...
0.17	9	0.156965	{0: 580, 1: 1169, 2: 663, 3: 1286, 4: 20, -1: ...
0.18	10	0.147659	{0: 589, 1: 1178, 2: 686, 3: 1301, 4: 25, -1: ...
0.19	10	0.141617	{0: 596, 1: 1196, 2: 692, 3: 1309, 4: 39, -1: ...
0.20	8	0.152674	{0: 618, 1: 2527, 2: 701, 3: 49, -1: 720, 4: 2...
0.21	1	-	{0: 4400, -1: 600}
0.22	1	-	{0: 4523, -1: 477}
0.23	2	0.337710	{0: 4642, -1: 348, 1: 10}
0.24	3	0.286715	{0: 4710, -1: 265, 2: 8, 1: 17}
0.25	3	0.279600	{0: 4784, -1: 191, 2: 8, 1: 17}
0.26	3	0.274118	{0: 4834, -1: 138, 2: 11, 1: 17}
0.27	2	0.313955	{0: 4874, -1: 90, 1: 36}
0.28	2	0.141891	{0: 4942, -1: 55, 1: 3}
0.29	1	-	{0: 4960, -1: 40}

3.3 Hiearchical clustering

Choice of attributes and distance function We have used the same attributes and distance function as for k-means and DBScan. We have rescaled all features into $[0, 1]$ range.

Dendograms We have generated dendograms for different algorithms. Except for single-linkage, they all look very similar. See the Figures 7a-7e.

The colors indicate the default suggestion by scikit-learn to cut the dendogram. It is at the 70% height of the dendogram. In the single-linkage case (figure 7a) we get two big clusters (green and red) and a few smaller clusters (blue). In all other cases we get just two clusters (green and red).

Note also, that we display only the top 30 levels of the dendograms.

Table 5: Distribution of examples in clusters

Cluster	# of points
noise	2208
0	1813
1	3623
2	2281
3	3823
4	95
5	680
6	237
7	129
8	80
9	30

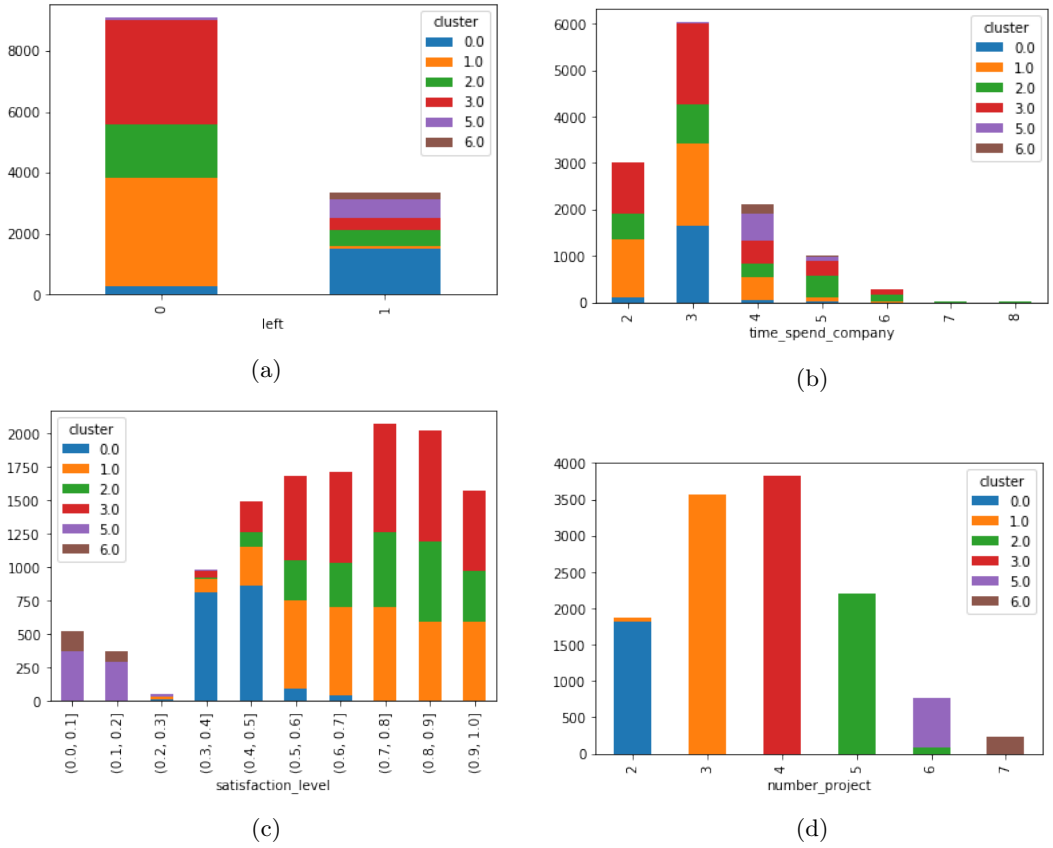


Fig. 6: (a):Cluster label rate by *left* classes, (b): Cluster label rate by *time_spend_company*, (c): Cluster label rate by *satisfaction_level*, (d): Cluster label rate by *number_project*

4 Association Rules

In this part, the association rules are represented from the data set, but firstly, the data set needs to be reprocessed and item sets to be found. To find the sets of items, the Apriori algorithm is executed with PyFIM [2] a python library. In addition, this library also mentions about extracting rules from these sets of items based on parameters such as: minimum confidence value (MinConf), minimum support value (MinSupp) and the minimum number in each frequent item set.

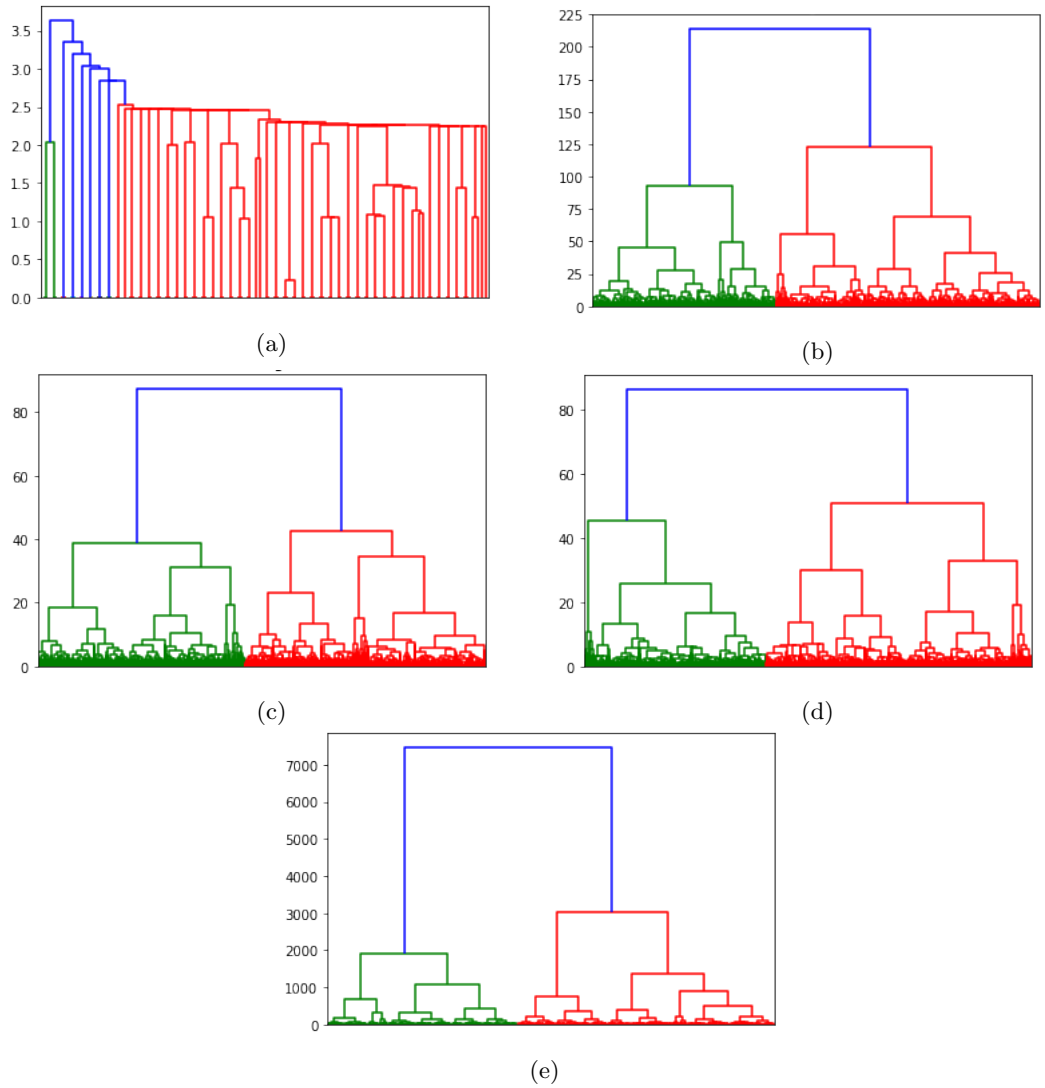


Fig. 7: (a): Dendogram obtained by Single Linkage hiearchical clustering, (b): Dendogram obtained by Complete Linkage hiearchical clustering, (c): Dendogram obtained by Average Linkage hiearchical clustering, (d): Dendogram obtained by Centroid hiearchical clustering, (e): Dendogram obtained by Ward hiearchical clustering

4.1 Preprocessing Data set

In this step, some attributes of the data set with continuous values need to be divided into bins because they have to be discrete data which can be applied the Apriori algorithm. Particularly, the "satisfaction_level" feature is divided into 10 bins with the range from 0 to 100. However, these values need to be converted to integer before dividing into bins. In case of the "last_evaluation" attribute, it should be done the same as the "satisfaction_level" attribute, whereas the "number_projects" is distributed in 6 bins with range from 2 to 7 (from the minimum to maximum values). Another attribute being the "average_montly_hours" is divided into 9 bins with range from 90 to 330 because every month is considered to have 30 days. The last attribute is the "time_spend_company" which is divided into 9 bins from 2 to 10 which ranges from the minimum value to the maximum value as well. Finally, these values are added information from its attribute.

4.2 Frequent Patterns Extraction

In frequent pattern extraction, the report considers not only the frequent item sets but also maximal and closed item sets. These item sets are considered and extracted by Apriori algorithm with PyFIM library from Python, but it needs some parameters such as: the MinSupp and the minimum number of items per item set.

With the Apriori function, the list of frequent, closed and maximal item sets can be extracted based on the different parameters such as: “a”, “c”, “m”.
However, the most interesting frequency item sets are maximal and closed item sets, because these kinds of item sets have the highest number of records or the highest elements in item sets, in turn. With this reason, the report considers interesting frequency item set being closed item sets, seeing as the maximal item sets are subsets of closed item sets. In addition, the item sets with the high MinSupp are generally more interesting than item set with low MinSupp, because we get a large number of records in data set.
Nevertheless, it depends on the meaning of item sets. For example, although the highest number of records in the Table 6 is “Non Accident” and “Non Promotion in 5 years” (it means that employees who do not have accident and do not have any promotion in 5 years), this item set does not have any meaning. From the Table 6, two items (“NonAcci” and “NonPro5Y”) appear together in almost the data set (12586 in 14998 records). Therefore, these values can be not too much interesting in the whole data set.
From the observation, the closed item sets with MinSupp being 30 in total data set are (“lowSala”, “NonL”), (“3TimeCom”, “NonL”), and (“mediumSala”, “NonL”) which are the most frequency item set in the data set (in case of removing “Work_accident” and “promotion_last_5years” attributes). The number of closed item sets is increased with lower MinSupp coefficients.

Table 6: The most interesting frequent item sets (Closed item sets) with different MinSupp

MinSupp	The number of Records	Item sets
40	12586	NonAcci, NonPro5Y
30	5144	lowSala, NonL
30	5129	mediumSala, NonL
30	4857	3TimeCom, NonL
25	3983	3NoPro, NonL
25	3956	4NoPro, NonL
20	3204	3TimeCom, lowSala
20	3191	2TimeCom, NonL
20	3126	salesSale, NonL
15	2718	3TimeCom, mediumSala
15	2390	150AverHour, NonL
15	2257	180AverHour, NonL
10	2219	3TimeCom, lowSala, NonL
10	2191	60LastEva, NonL
10	2181	240AverHour, NonL
10	2171	L, lowSala
10	2164	3TimeCom, mediumSala, NonL
10	2155	50LastEva, NonL
10	2149	5NoPro, NonL
10	2132	210AverHour, NonL
10	2105	70LastEva, NonL
10	2099	salesSale, lowSala
10	2087	4NoPro, lowSala
10	2023	technicalSale, NonL
10	2000	Acci, NonL
...	...	...

4.3 Extracting Association Rules

Association rules are extracted from item sets. Therefore, there are two parameters: MinSupp and MinConf coefficients using for rule extraction.
Because of many extracted rules, the lift value is used to remove rules whose head and body are independent to each other. That means if a rule has the lift value being greater than one, this rule will be useful, otherwise it will be useless.
After rules extraction, the Tables 7, 8, 9 and 10 illustrated the set of rules with different coefficients (the MinSupp and MinConf coefficients). In more detail, the first column represents the MinSupp coefficient of rules, while the percentage of confident coefficient of rules is showed by the second column. The third illustrates the lift values, and the remaining columns are head and body of rules, respectively.

It can be clear that these tables are sorted based on the lift values, because the lift value, one of measure, is used to evaluate the usefulness of rules. With this reason, the lift value of rules being less than one is removed in these tables. Moreover, the table with low MinSupp or MinConf exists overlap rules from the tables which have high MinSupp or MinConf. Therefore, the overlap rules are deleted.
In addition, the rules whose head are the same and the body is the subset of each other it can be eliminated because of overlap. For example, if the rule "A ← B, C" has the lift value 1.8, but the rule "A ← B" has the lift value 1.9, which means the later covers the other, the rule "A ← B, C" will be removed.
The extracted rules from these tables can be said that employees who did not leave the job have the high values in "satisfaction_level", "average_montly_hours", "last_evaluation" and "salary" attribute (the high values from 50 to 90 in "satisfaction_level" attribute, from 150 to 210 in "average_montly_hours" attribute, from 60 to 70 in "last_evaluation" attribute and high or medium in "salary" attribute). In addition, the "number_project" and "time_spend_company" attributes effect to the decision of employees in left or non left the job. In particularly, the employees who have 3 "time_spend_company" and 3 or 4 "number_project" will not leave the job, while the employees will leave the job in case of 2 "number_project" and 3 "time_spend_company". On the other hand, it can be said that employees who left the job can imply 2 "number_project" and 3 "time_spend_company".

Table 7: Rules are extracted from the item sets with MinSupp and MinConf being 10 and 90

MinSupp	MinConf	Lift	Head	Body
10	96.496%	6.063	2NoPro	L, 3TimeCom, NonPro5Y
10	96.340%	6.053	2NoPro	L, 3TimeCom
10	97.509%	2.270	3TimeCom	2NoPro, L
10	99.494%	1.305	NonL	3NoPro, 3TimeCom
10	98.788%	1.296	NonL	4NoPro, 3TimeCom, NonPro5Y
10	98.665%	1.294	NonL	4NoPro, 3TimeCom
10	98.559%	1.293	NonL	2TimeCom, lowSala
10	98.367%	1.290	NonL	60Satis
10	98.366%	1.290	NonL	2TimeCom
10	98.224%	1.289	NonL	3NoPro
10	97.917%	1.285	NonL	180AverHour
10	97.725%	1.282	NonL	60LastEva
10	97.491%	1.279	NonL	50Satis
10	93.924%	1.232	NonL	90Satis
10	92.527%	1.214	NonL	70LastEva
10	92.208%	1.210	NonL	Acci
10	91.727%	1.203	NonL	4NoPro, mediumSala
10	90.630%	1.189	NonL	4NoPro

Table 8: Rules are extracted from the item sets with MinSupp and MinConf being 10 and 80-90

MinSupp	MinConf	Lift	Head	Body
10	84.305%	3.541	L	2NoPro, 3TimeCom, NonAcci, NonPro5Y
10	84.241%	3.539	L	2NoPro, 3TimeCom, NonAcci
10	82.426%	3.462	L	2NoPro, 3TimeCom, NonPro5Y
10	82.406%	3.462	L	2NoPro, 3TimeCom
10	85.382%	1.120	NonL	210AverHour
10	83.940%	1.101	NonL	70Satis

Table 9: Rules are extracted from the item sets with MinSupp and MinConf being 8-10 and 90

MinSupp	MinConf	Lift	Head	Body
8	99.123%	1.300	NonL	2TimeCom, 3NoPro
8	93.371%	1.225	NonL	highSala

Table 10: Rules are extracted from the item sets with MinSupp and MinConf being 8-10 and 80-90

MinSupp	MinConf	Lift	Head	Body
8	81.810%	5.140	2NoPro	40Satis, 3TimeCom
8	83.593%	1.946	3TimeCom	2NoPro, lowSala, NonAcci, NonPro5Y
8	83.387%	1.941	3TimeCom	2NoPro, lowSala, NonAcci
8	81.292%	1.892	3TimeCom	2NoPro, lowSala, NonPro5Y
8	81.161%	1.889	3TimeCom	2NoPro, lowSala
8	84.814%	1.113	NonL	150AverHour, mediumSala

4.4 Using the Meaningful Rules for Prediction

From extracted rules, missing values can be predicted instead of using some other formulas such as: mean or average to fill the missing values. Fortunately, in this data set, the missing values do not exist; however, in case of existed missing values, these rules can be utilized.

Table 11: The accuracy of using the interesting rules to predict the data set.

Parameters	The number of accuracy	Accuracy
MinSupp: 10 and MinConf: 90	10762	0.717
MinSupp: 10 and MinConf: 80	12709	0.847
MinSupp: 08 and MinConf: 90	10837	0.722
MinSupp: 08 and MinConf: 80	12740	0.849
MinSupp: 05 and MinConf: 90	12509	0.834
MinSupp: 05 and MinConf: 80	13527	0.901
MinSupp: 10 and MinConf: 90	3233	0.71
MinSupp: 10 and MinConf: 80	3830	0.851
MinSupp: 08 and MinConf: 90	3254	0.723
MinSupp: 08 and MinConf: 80	3840	0.853

The first part of the Table 11 represents the accuracy of using rules on the whole data set. It is clear that if using a low MinSupp and high MinConf coefficients to predict the “left” attribute, the accuracy is higher than using high MinSupp and low MinConf parameters. The reason is that with lower MinSupp and high MinConf, there are many extracted rules, which leads to the prediction of the variety of situations. Therefore, the accuracy can be higher than using the high MinSupp and low MinConf.
On the other hand, the remaining part of the Table 11 shows the accuracy of using extracted rules for prediction employees in case of splitting the data set into two parts including training set (70%) and testing set (30%). In more detail, the training set is randomly combined 70% records of “Non Left” and “Left” employees, while the remaining data set is the testing set. When applying the extracted rules to predict the test set, the accuracy of prediction is quite similar when applying these rules to the total data set.

5 Classification

Classification part of the project contains preprocessing of data to normalize values in order to gain real results, distinguish dataset to train and test set and after that building the model. We used different criterions and approaches to compare and choose the best scores.

5.1 Preprocessing of dataset, selecting feature to predict, partitioning into training- and test set

At the beginning we converted object values to numerical ones. In this case they were chosen attribute sales and salary. Later we considered reshaping the values of attributes, which were not nominal and the values were not between 0 and 1 i.e. number_project, average_montly_hours and time_spend_company. All of these operations were applied to systematize the data to obtain correct results. At the end we converted the form of data from DataFrame to a numpy array. Left attribute was chosen to be predicted in our project. Based on this we cleaved

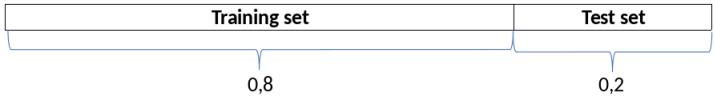
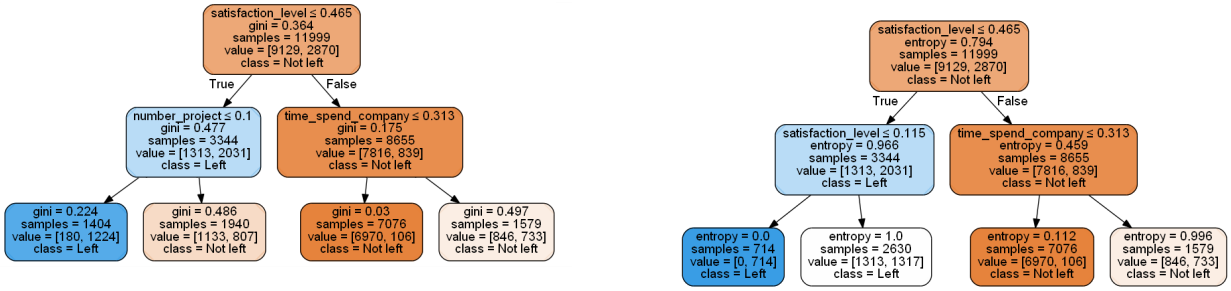


Fig. 8: Partitioning of training and test set



(a) (b)

Fig. 9: (a): Decision Tree with criterion gini, (b): Decision Tree with criterion entropy

off two arrays: features and target to construct and fit the model. We decided to distinguish training and test set with proportion 80% of training set and the rest, 20% of test set(Figure 8).
It was decided to use two kind of models: Decision Tree Classifier and Random Forest in all of them we considered two different parameters as criterion: gini and entropy.
Formula for the gini index for a given node t:

- $Gini(t) = 1 - \sum_j [p(j \vee t)]^2$
- $Entropy(t) = -\sum_j p(j \vee t) logp(j \vee t)$

5.2 Decision Tree Classifier

At the beginning we decided to use DecisionTreeClassifier as a model and based on it build the classification tree. Consider the undermentioned figures of trees we should stress that in general they were built in the very similar way to eliminate the kind of measures of node impurity, so they will be described together with explanations on their differences (Figure 9a and 9b).
In the first node, root, we have in both cases "satisfaction_level" as an attribute, which one starts distinguishing the data to predict. If the value of this attribute is lower or equal to 0.465 we can go to the left node, in compliance with True in the arrow. Otherwise we choose the right side of the tree. In the root there is also performed how many samples we have and how the are distribution- list of value (to sum up elements of the list we should receive the number of samples). Every next node has different attributes, which is the splitter of data. Refer to gini and entropy the value of them gives an information about the distribution of the data to the specified cases. It is the most interesting when the value of them is equal 0, because in this case we have clearly separated data. We can stop build the tree when all of the records belong to the same class or when the records have similar attribute value.
After the predicting, it was check how good is our model- we did this in two ways, first with training set, after that with test set- it is very useful to check if we do not have an overfitting (the overfitted model contains more parameters than can be justified by the data). We can suspect overfitting when the difference between accuracy with training and test set is very large. In our case, we can see that we dont have the problem of overfitting (Table 12). As we can see the results are better given the criterion gini, but the difference is very small. To present the tree we decided to use with max_depth=2, min_samples_split=2, min_samples_leaf=10. We are aware that in this situation we have underfitting, but we decided to consider this case, because of putting an image of the tree and describe it.
It was also counted how good is our model. In order to do that we used the formulas, which are represented below- the scores are based on test set (Table 13). E.g. accuracy tells us how accurately our model predicted the target value. As beforehand we achieved better scores with gini.
To visualize the quality of the model and prediction there are a lot of ways available, we chose a few of them. The first one is confusion matrix, which considers all four opportunities of predicting (Table 14), (Figure 10a and 10b). As we can see, with gini, we got better results than with entropy. With gini the sum of TP and TN is

Table 12: Accuracy of training- and test set because of gini and entropy criterions

Parameters	max_depth=2, min_samples_split=2, min_samples_leaf=10		max_depth=5, min_samples_split=2, min_samples_leaf=10		max_depth=8, min_samples_split=2, min_samples_leaf=10		max_depth=15, min_samples_split=2, min_samples_leaf=10	
Criterion	Gini	Entropy	Gini	Entropy	Gini	Entropy	Gini	Entropy
Training Accuracy	0.847	0.82	0.973	0.972	0.977	0.978	0.979	0.979
Testing Accuracy	0.858	0.82	0.973	0.972	0.976	0.977	0.974	0.974

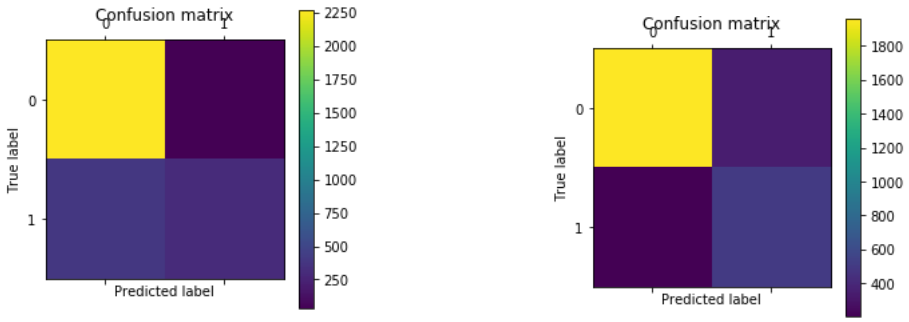
Table 13: Measures of precision, recall, F1 and accuracy because of gini and entropy

Parameters	max_depth=2, min_samples_split=2, min_samples_leaf=10	
Criterion	Gini	Entropy
Precision	0.865	0.834
Recall	0.858	0.820
F1 measure	0.838	0.825
Accuracy	0.858	0.820

higher than with entropy by 115. This approach was used on the test set. After that it was applied the measures, which are on the table (Table 15). Also in this case gini is better than entropy.

Table 14: Confusion matrix of prediction.

Actual class	Predicted class				
		Gini		Entropy	
		Left	Non Left	Left	Non Left
	Left	TP=2268	FN=31	TP=1960	FN=339
	Non Left	FP=394	TN=307	FP=201	TN=500



(a) (b)

Fig. 10: (a): Image of confusion matrix for criterion Gini, (b): Image of confusion matrix for criterion Entropy.

To validate our model we used also another methods.
Cross validation method of model validation, where we change the training and test set in every round in order to cover every combination (number of combination sets k-fold) of different partitions (Table 16). Even with cross-validation there is still a difference between gini and entropy. Excluding precision, with CV we got

Table 15: Comparison of measures of predicted classes

	Precision		Recall		F1		Support	
Criterion	Gini	Entropy	Gini	Entropy	Gini	Entropy	Gini	Entropy
Non Left	0.85	0.91	0.99	0.85	0.91	0.88	2299	2299
Left	0.91	0.60	0.44	0.71	0.59	0.65	701	701
Avg/Total	0.87	0.83	0.86	0.82	0.84	0.83	3000	3000

better results with entropy, but also the difference in precision is very small. About the values of recall and F1 measure, our data is so unbalanced and partition of data with k-fold was randomly, so it is possible that in one combination we could have values of attribute left equal to 0. As k-fold we assumed cv=10.

Table 16: Comparison of precision, recall, F1 measure and accuracy- decision tree-cross validation with gini and entropy criterions

Classifier	Decision Tree	
Parameters	cv=10, max_depth=2, min_samples_split=2, min_samples_leaf=10	
Method of Validation	Cross Validation	
Criterion	Gini	Entropy
Precision	0.79 (+/- 0.35)	0.82 (+/- 0.51)
Recall	0.48 (+/- 0.22)	0.36 (+/- 0.31)
F1 measure	0.57 (+/- 0.05)	0.44 (+/- 0.16)
Accuracy	0.82 (+/-0.10)	0.79 (+/- 0.11)

5.3 Random Forest

It was decided to use another classifier with our data- Random Forest Classifier. It is a method for classification or regression with construction multitude of decision trees of the random data. To distinguish it with normal decision tree classier, Random forest is not liable to overfitting. To validate we also used Cross Validation (Table 17). What is very interesting, with Random Forest for the first time there is no difference between gini and entropy criterion (the difference between the values of recall is very small, so we can assume that they are almost equal). About the parameters we used n_estimators=30, it means that we have 30 trees in our forest, like m_features we assumed number of features in our data (max_features=None), max_depth=None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples, bootstrap=True, which means that we used bootstrap samples (bootstrap means with minimalresources), random_state= None, it means the random number generator is the RandomState instance used by np.random. and class_weight=None- all of the classes have the same weight.

Table 17: Comparison of precision, recall, F1 measure and accuracy- Random forest-cross validation with gini and entropy criterions

Classifier	Random forest	
Parameters	cv=10, n_estimators=30, max_features=None, max_depth=None, min_samples_split=2, min_samples_leaf=5, bootstrap=True, oob_score=False, random_state=None, class_weight=None	
Method of Validation	Cross Validation	
Criterion	Gini	Entropy
Precision	0.99 (+/- 0.02)	0.99 (+/- 0.02)
Recall	0.93 (+/- 0.03)	0.94 (+/- 0.03)
F1 measure	0.96 (+/- 0.02)	0.96 (+/- 0.02)
Accuracy	0.98 (+/- 0.01)	0.98 (+/- 0.01)

5.4 Automatic parameter selection, search the best classifier

Automatic parameter selection could be very useful when we want to find the best model, to check every combination manual is impossible, thats way two approaches described below became indispensable.
In this part we compared two approaches: RandomizedSearchCV and GridSearchCV for optimizing hyperparameters of a random forest. As a method of validation we used Cross Validation.
RandomizedSearchCV provides a fit method and also the score. In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The parameter n_iter tells us how many times the parameters are tried, so when GridSearchCV can be computationally expensive, because exhaustively generates candidates from a grid of parameter values specified with the param_grid, RandomizedSearchCV is the best solution. About the parameters we can add the list of them. The parameters of GridSearchCV is a list values of max_depth, whereas the parameters of RandomizedSearchCV with 3-fold in CV, 100 iterations and 10 n_job is followed:

- Max_depth: [2,3,4,5,6,7,8,9,10,11,12,None]
- Max_features: Random int between (1, number of columns-1)
- Min_samples_split: Random int between 1 and 51
- Min_samples_leaf: Random int between 1 and 51
- Criterion: Gini, entropy

For RandomizedSearchCV we described the full list of parameters, because in this case we can set n_iter. With GridSearchCV we set only max_depth, because of exhausting of this approach (in this case all combinations of the parameters could be checked, what is very expensive). According to RandomizedSearchCV and GridSerachCV five the best models (Table 18 and 19).

Table 18: Five the best models- RandomizedSearchCV

	Accuracy	Max_depth	Max_feature	Criterion
1	0.974 (+/-0.003)	10	6	Entropy
2	0.974 (+/-0.003)	12	6	Entropy
3	0.973 (+/-0.002)	12	2	Gini
4	0.973 (+/-0.002)	8	3	Gini
5	0.973 (+/-0.002)	11	3	Entropy

Table 19: Five the best GridSearchCV

	Accuracy	Max_depth
1	0.991 (+/-0.003)	None
2	0.985 (+/-0.002)	12
3	0.983 (+/-0.002)	11
4	0.985 (+/-0.002)	12
5	0.985 (+/-0.002)	12

Analyzing foregoing tables, we can see that the best is the first model generated of GridSearchCV. In this case we have max_depth=None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

6 Conclusion

This assignment is the implementation of tasks (DU, clustering, AR and classification) on the data set of Human Resources. The implementation of tasks are implemented with different and various techniques for each task. Particularly, the DU part exploits general information such as mean, min, max values of each attributes. The clustering part uses the K-means, DBScan and Hiearchical and AR part utilize the Apriori algorithm for frequency item sets and extracted rules. Meanwhile, the classification is applied with decision tree and random forest combining cross-validation and grid search. As a result of our analysis we should mention, that the variable *satisfaction_level* as well as *time_spend_company* strongly influences the decision of the employees to leave the company. This was suggested in during the clustering phase of the project and later validated by the association rules.

7 Ackowledgements

We would like to show my gratitude to Prof. Dino Pedreschi, Prof. Anna Monreale, Dr. Riccardo Guidotti about theory lectures and implementation in class.

References

1. Ludovic benistant. Data set: HR Analytics. `https://www.kaggle.com/ludobenistant/hr-analytics-1/data/`. [Online; accessed 04 Jan 2018].
2. Christian Borgelt. Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):437–456, 2012.