IBM HR Analytics Employee Attrition & Performance

Blerta Lleshi
Claudio Monti
Giovanni Scognamiglio
Agnese Simonelli

Introduction

Employee attrition is a major issue in the companies' day to day human resource management. It is costly and can have negative long term consequences; however, firms can address attrition in multiple ways. Organic traditional methods include improving the selection process to ensure cultural fit, monitor employees satisfaction metrics and enhance retention plans.
The purpose of this paper, a class related project for Data Mining (A.Y. 2020/2021) course at Università di Pisa, is to go beyond the traditional methods to cope with attrition and use modern data mining techniques to obtain actionable insights for significantly improving the efficiency of those traditional methods.
We will first try to discover hidden patterns that could for instance help the hiring selection process by identifying which candidate characteristics result in lower attrition. We will then try to detect association rules that would provide firms more information on the true reasons why employees leave the company, enabling firms to monitor the right metrics and act upon them.
Finally, we will try to build a classification model which would predict whether an employee will leave based on his characteristics, such model would allow firms to identify the valuable employees which have higher probability of quitting and offer them an enticing tailored retention plan.

Chapter 1 Data Understanding

1.1 Data Semantics

The object of our study is the *'Train_HR_Employee_Attrition'* dataset, processed using Python. The dataset consists of 1176 sample records and each of them can be framed as a collection of attributes that include several predictors for detecting employee voluntary attrition tendency. The objects refer to individual employees and attributes that range from objects' age, monthly income and job satisfaction. The target variable is *Attrition*, which indicates whether the object left the company. All 33 attributes appear to be related to employee attrition and selected attributes with detailed description are presented in a tabular form as follows:

Variable Name	**Category**	**Description**	**Values**
Attrition	Binary	Voluntary reduction of employees	{Yes, No}
Hourly/Daily/Monthly Rate	Numerical Continuous	Cost for an employee's services for a single day	-
Job Level	Categorical Ordinal	Job level of employee	[1,5]
Over Time	Binary	The employee works overtime	{Yes, No}
Stock Option Level	Categorical Ordinal	Equity compensation granted to employee	[0,3]

1.2 Data Distribution

The distribution of the target variable is highly imbalanced. Out of the 1176 employees in the population, individuals unwilling to prematurely leave the company, on a voluntary basis, comprise the largest group with 83.7%. Only 192 individuals (16.3%) have a positive attrition.

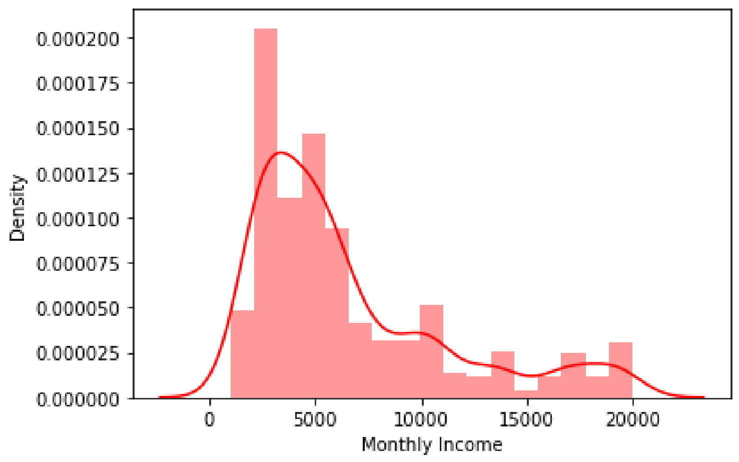


Figure 1: Distribution of Monthly Income

Figure 1 is representative of all the monthly incomes which were included within the dataset provided. It yields a graphical representation of grouped frequency distribution and is also useful in determining the distribution of the data.
From Figure 1, it can be seen that the data ranges from $ 1000-$ 20000. The income distribution is positively right skewed because a larger percentage of the incomes are located on the lower tail. The portion of employees with a salary around $ 20000 is mostly responsible for the histogram being right-skewed. Indeed, the mean ($ 6565.94) is greater than the median ($ 4969), because the mean is sensitive to the large salary of around $ 20000 and is pulled in the direction of the unusually large observations. The spikes of the distribution of incomes are clustered between $ 2500-$ 5000. Above this level of salary, the density of distribution is less affected.

We explored the distribution of monthly income with respect to:

- Gender; Education field; Job role; Business travel; Departments.

From Figure 2a, we can see that males and females tend to have fairly closely different monthly earnings profiles; more females tend to be in higher pay brackets, earning around $ 6’700 compared to males’ average of $ 6’500.
While the distribution of income by gender provides useful knowledge on the pay difference between males and females, the education field is also an important factor to identify the most paid category of employees based on their education.

Comparing monthly income by education field, Figure 2b shows that employees working in qualified fields such as *Medical*, *Technical Degree* and *Other* are strongly clustered around an average of monthly income of $ 7121.83. It can be interpreted that the employees belonging to these educational fields are more prone to have a higher monthly income in comparison to other education fields (i.e. employees with academic background in *Marketing* have a monthly income of $ 5624.85).

The job role is an important metric to track the employees’ monthly income. Figure 2b illustrates that a greater share of employees holding the position of Research Director tend to dominate the higher end of the monthly income distribution compared to those holding the position of Manager. Our analysis outlines that the difference, in monthly compensation, between these roles, is moderate and close to 26.5%.

To get a complete picture of employees monthly income we have also considered Business Travel’s influence.

From Figure 2a, it can be observed that the distribution of income is almost uniform in all the three event types (travel rarely, travel frequently and non-travel). Those who travel rarely have a comparative marginal advantage in terms of monthly income compare to those who travel frequently or those who have never travelled.

The department where the employees are allocated influences the monthly income. The Figure 2a, shows that the average income for the three departments included in our analysis is close to $ 6529.8.

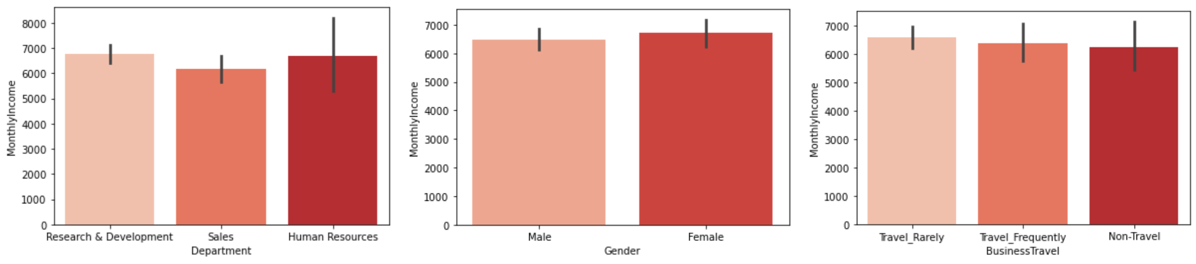


Figure 2a: Distribution of Monthly Income by different selected attributes

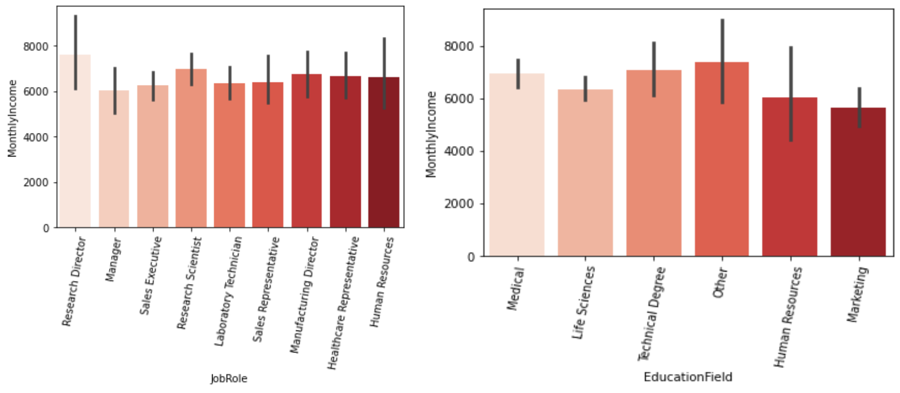


Figure 2b:Distribution of Monthly Income by different selected attributes

From Figure 3, we can see that employees in the R&D department have the highest attrition number but it's the Human Resources department that has the highest attrition rate (21%). We also noticed a tendency for young employees to exhibit a slightly higher attrition rate. One interpretation could be that younger employees hold different values about what is important in work and life.
An interesting insight is drawn from plotting satisfaction levels, indeed as one would expect, a satisfied employee has a lower attrition rate. Finally, employees working as Sales Representative have the highest chance of attrition compared to other job roles.

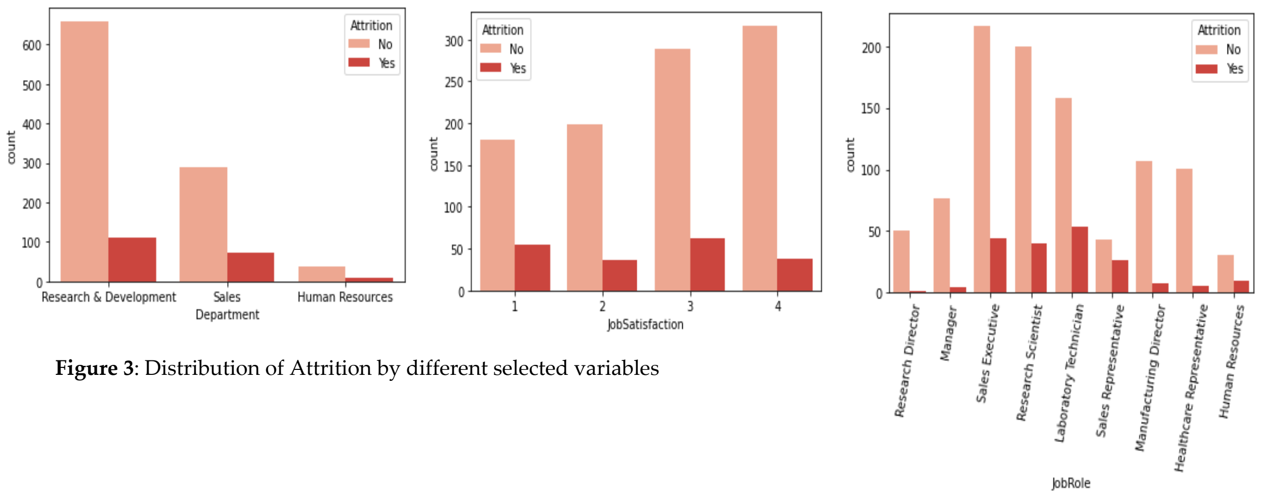


Figure 3: Distribution of Attrition by different selected variables

1.3 Data Quality

We first checked for label inconsistency and have identified that attributes *StandardHours* and *Over18* have a single value; therefore, we decided to drop them. Except from attribute *Over18*, no inconsistent value labels were found for attributes with character measurement scale.

Regarding variables with numeric measurement scale, we identified various problems. From Figure 4, the variable *TotalWorkingYears* appears to have inconsistent values. Indeed, when compared with employees' *Age* attribute, it suggests that 12% of employees started working before they were 15 years old and 2% started working even before they were born (having more work experience than years of age). We also compared it with the attribute *YearsAtCompany* and found similar inconsistent results (27% of employees have worked at the company longer than their total working years). We then plotted *YearsAtCompany* against *Age* and obtained a consistent result, thus assuring us that the inconsistent variable is indeed *TotalWorkingYears*. As the distribution of *TotalWorkingYears* and *YearsAtCompany* are similar, we deduced that the inconsistency of the former is probably due to a measurement error and we thus tried to interpret the variable in other possible ways. However, we have obtained no good results and decided to drop the entire *TotalWorkingYears* attribute.

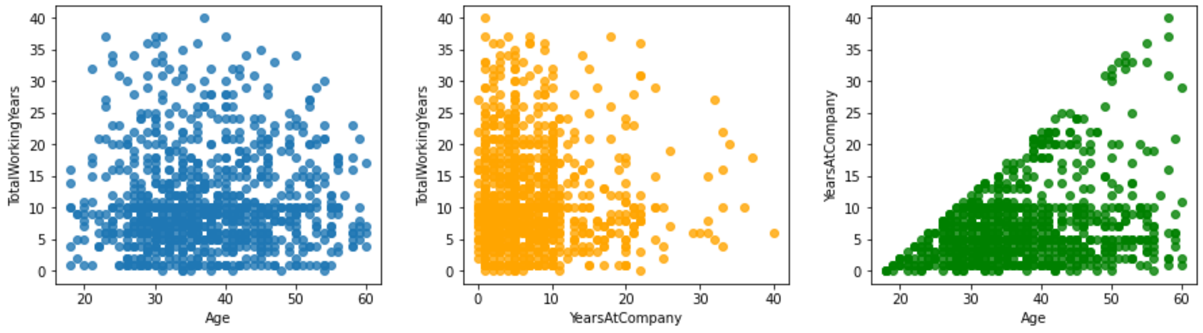


Figure 4: Values consistency check of *TotalWorkingYears*

Furthermore, we also noticed values inconsistency in the following variables: *YearsInCurrentRole*, *YearsSinceLastPromotion*, *YearsWithCurrManager*. Once again, inconsistency arises at time level as more than 36% of objects have a higher value for one of those variables than the *YearsAtCompany* variable; shown in Figure. (eg. some employees spent 10+ years with their current manager but only 1 year at the company)

We have also tried to see whether these inconsistent features were due to errors in measurement scale; however, we could find no rational explanation or transformation to include those variables in our further analysis and therefore decided to drop these inconsistent variables.

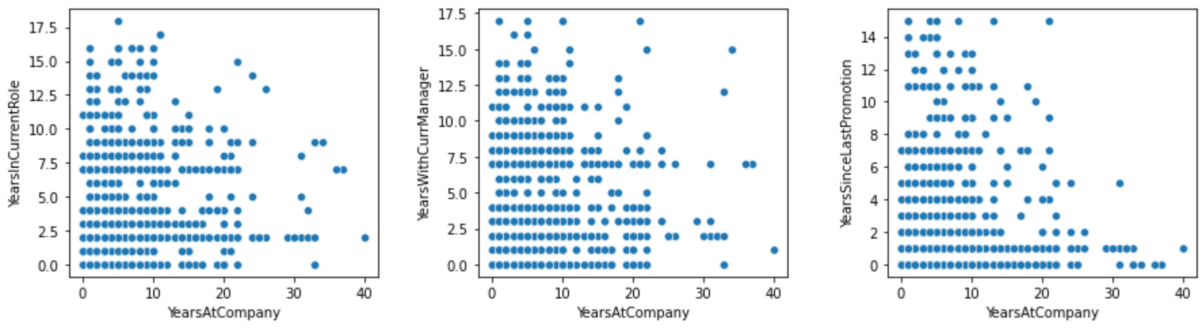


Figure 5: Plot of the difference between *YearsAtCompany and YearsInCurrentRole*

Regarding missing values, we identified various variables with missing values (Figure 6) and summarized their computation as follows:

- ❖ *Gender* (59 missing values): we replaced missing values with the mode for a given *JobRole* and *MaritalStatus* (as they were the best variables without missing values explaining *Gender*).

- ❖ *YearsAtCompany* (60 missing values): we first tried a least square regression approach using variables such as *MonthlyIncome* and *Age*; however, the fit was not good. We then filled the missing values with the mean of *YearsAtCompany* when grouped by *JobRole, Gender* and *MaritalStatus*.
- ❖ *BusinessTravel* (107 missing values): Using the mode, even when aggregated from other variables, was not providing good results as the variable is unbalanced and its distribution was constant among all other categorical variables. We therefore replaced missing values with randomly generated values taking into considerations the weights of the original distribution.
- ❖ *PerformanceRating* (138 missing values): filled with resulting mode for a given *Education* and *JobLevel*.
- ❖ *Age* (176 missing): imputed with mean for given *Gender, JobSatisfaction, EnvironmentSatisfaction* and *StockOptionLevel*.
- ❖ *MonthlyIncome* (213 missing) : imputed with mean for a given *Gender* and *Age* category (a new column we created from Age).
- ❖ *TrainingTimesLastYear* (233 missing): we decided to drop this variable. It's distribution is mostly centered around two values and it appears to have no real relevance to the attrition problem.

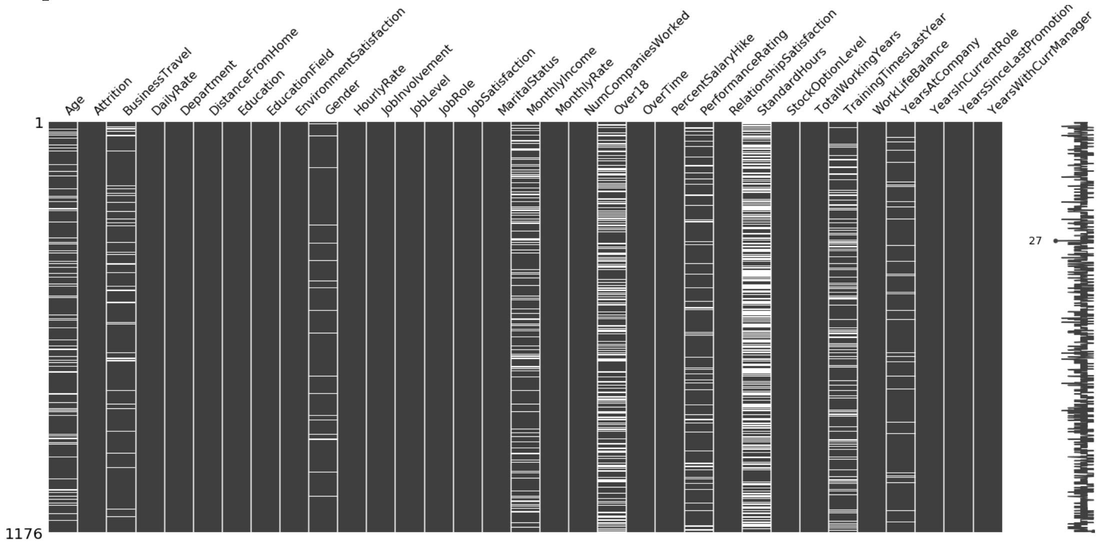


Figure 6: Missing Values

After each variable imputation, we checked the new distribution of the variable and asserted that the new distribution was almost or acceptably similar to the pre-modified one.

1.4 Variables Transformation

Concerning the task of *Variables Transformation*, we made different operations for various values in order to obtain a better distribution, a simplification and a preparation for the later operations of clustering, pattern mining, classification and so on.

In order to help us with statistical operations and to allow the algorithms to work better, we have started with a fundamental operation: normalization. We have scaled the variables distribution and prepared the data for the subsequent cluster analysis. The variables that we normalized were *MonthlyRate, DailyRate, HourlyRate* and *YearsAtCompany*.

We further opted to compute an operation of *Binarization* for variables such as *Attrition, OverTime* and *Gender*. As concerning the first two, with this transformation we have replaced Yes with 1 and No with 0. While for the third one we have replaced Male with 1 and Female with 0. We have reputed that this operation could be useful for eventual operations in the association analysis and to calculate the Jaccard index.

For the variable *MonthlyIncome*, we noticed that the distribution is right-skewed with a minimum of 1009 and a maximum of 19999. In order to make the distribution less asymmetric and better analyzable with statistical operations, this time we opted to transform it with a logarithmic transformation. After the log transformation, with the visualization of the scatter plot, the sparsity of the point has been reduced.

With regard to the attribute *DistanceFromHome*, we chose to discretise the variable. After calculating the max. value (=29) and the min. value (=1), we made a natural binning and we obtained four intervals with the same length size. In addition, we labelled each interval as follows:

- 8 < : *Proximate = 1*
- 8-15 km: *Near = 2*
- 15-22 km: *Intermediate = 3*
- 22-29 km: *Far = 4*

Instead, for attributes as *Age*, we proceeded with the *Discretization*, in which we have created different intervals, following the Sturge's rule, that divide the population in subsets. In our case we obtained 12 bins with an interval size that swings between 4 and 5 years.
For some variables we chose to divide in more numbers of bins, in order not to lose important information, in others, we considered to use less intervals because the result remained unchanged.

1.5 Correlation

To build a correlation matrix with all the attributes, we converted categorical variables into dummy/indicator variables through the *get_dummies* function.
The correlation matrix obtained after this transformation is too large to be shown, thus we have opted to annex it in appendix A.
We found out that generally all the attributes are weakly correlated, except for a few:

- ❖ *MonthlyIncome* is correlated with *Age* and *YearsAtCompany*.
- ❖ *JobLevel* is highly correlated with *TotalWorkingYears*.
- ❖ *YearsInCurrentRole*, *YearsSinceLastPromotion* and *YearsWithCurrent* manager are between them correlated, in particular the first and the third are highly correlated.
- ❖ *Department* is highly correlated with *JobRole*, in fact only certain working figures work in each department, as shown in Figure 7. For this reason we decided to eliminate the department attribute. The job role manager is present in each department but in order to not lose this information we replaced the value manager with three possible values *Manager_R&D*, *Manager_Sales* and *Manager_HR*.

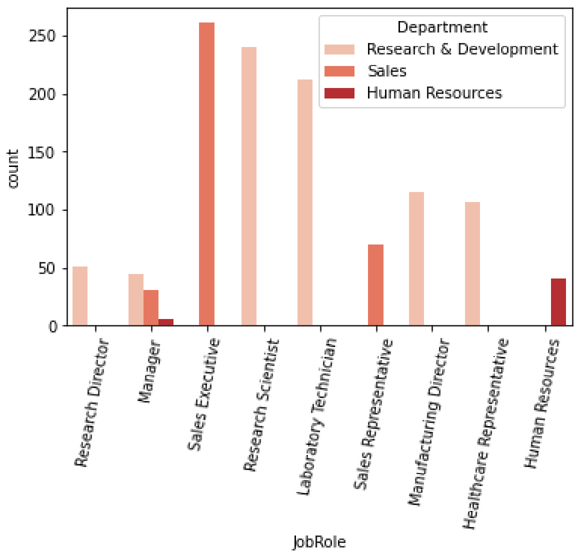


Figure 7: Distribution of roles over department

Chapter 2 Clustering

2.1 K-Means

In this section, we will present the results of the K-Means clustering algorithm application.

2.1.1 Feature scaling and Attributes selection

In our K-Means clustering analysis, we tried scaling the features both with MinMax normalization and the *z-score* standardization. We observed a consistent drop in resulting clustering SSE when features were normalized with the MinMax method and therefore we chose the latter one.

Considering the semantics of our dataset, we chose to use K-Means with variables we considered were mostly related to the attrition issue in a company. However, as we had to drop potentially interesting variables such *YeasAtCompany* and *YearsInCurrentRole* for inconsistencies (as reported in 1.3), we proceed with a single dataframe for our analysis composed of only the few interesting variables we had left: *Age, MonthlyIncome* and *YearsAtCompany*.

2.1.2 Identification of best k

We identified the best number of clusters for our dataframe with the Knee method by computing the SSE for k up to 20.
As shown in Figure 8, the MinMax normalization yielded much lower SSE than *z-score* normalization.

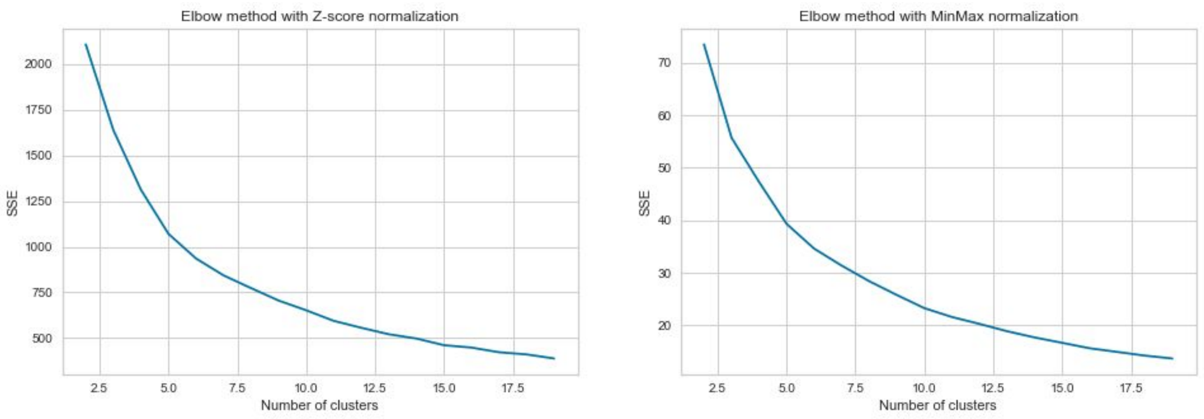


Figure 8: Comparison between z-score and MinMax scaler normalization on SSE

We have discovered, using the knee method, that the best number of clusters was k = 5. However, we decided to discard this result since two clusters resulted closely when plotted on a two dimensional plan. We therefore used k = 4 as the best number of clusters both from semantically and parameter wise point of view.

2.1.3 Description of the best clustering

We analysed the results of the clustering obtained in the previous section. We first plotted the clusters in a two dimensional plan and then provided a description of the core points of each cluster with respect to its semantic interpretation.
Figure 9 shows the clusters plotted on a two dimensional plan (*Age* and *MonthlyIncome).* We didn't include *YearsAtCompany* as two former variables yielded a better cluster visualization (centroids are marked with a star and clusters are identified with the legend on the top right).

We described the various clusters visible in Figure 9 and then we have provided a textual interpretation of the type of employee reflected in each cluster's centroid.
The centroids coordinates have to be interpreted as follows:

centroid = (*Age, MonthlyIncome, YearsAtCompany*)

- Cluster 0 with centroid (36 y, 6'781 $, 8.3 y): young adult employees with a medium monthly income who worked for the company for about 10 years;
- Cluster 1 with centroid (29 y, 3'533 $, 4 y): very young employees with a low monthly income who worked for the company for a few years;
- Cluster 2 with centroid (47 y, 14'738 $, 13 y): adult employees with an high monthly income who have worked for the company for several years;
- Cluster 3 with centroid (46 y, 4'798 $, 5 y): adult employees with a medium low monthly income who worked for the company for a few years.

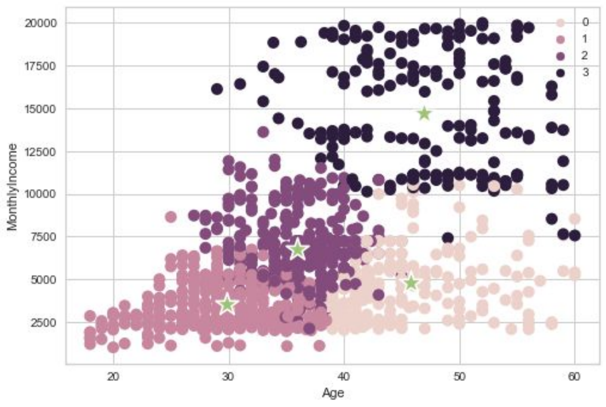


Figure 9: The four cluster with their centroids show as green stars

Given these clusters, we tried to understand whether any of these groups presented a significant variation regarding employee attrition. We first applied the clusters labels to our dataset and then plotted a normalized cross tabulation of the clusters labels with the target variable *Attrition*. As shown in Figure 10, the dataset objects were most commonly associated with cluster 1 (young employees with low income), followed by cluster 0 (young adults with medium income). However, as we can see from the subplot *"% rate by attrition"* no clear disparity between the clusters regarding the *Attrition* was visible.

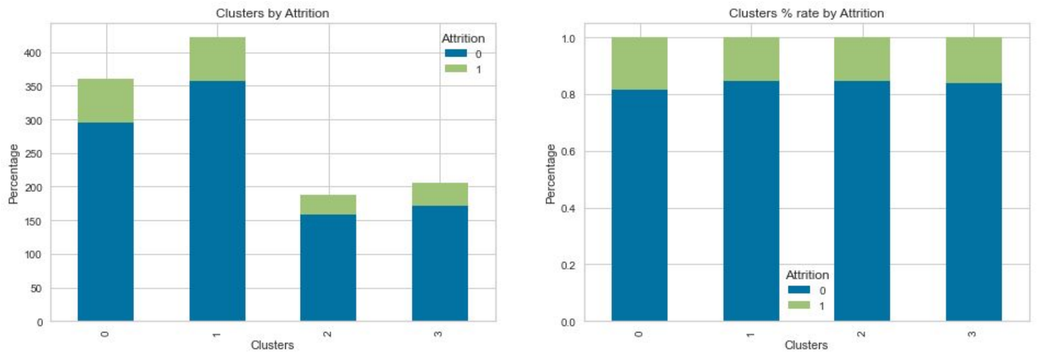


Figure 10: **a**) The distribution of employees over the four clusters; **b**) The percentage of attrition in each cluster

Therefore, we concluded that there was not a significant difference of employee attrition among the various clusters we have identified with the K-Means algorithm.

2.2 DBSCAN

In this section, we will discuss Density-based cluster analysis. The first part is dedicated to the selection of the optimal variables that will be used in the algorithm, explaining the reasons behind those choices. In the second part, we will present some considerations regarding the selection of the following important parameters:

- MinPoints parameter (MinPts);

- ❖ Epsilon parameter (Eps).

In the final part, we will comment on the outputs that we have obtained. We will compare the results with other clusters' results, in order to examine which is the most appropriate for our data frame.

2.2.1 Choice of attributes and distance function

For what concerns the selection of attributes, we opted to choose those that are numerical, which have been normalized and transformed in Section 1.4. In particular, after the creation of a heatmap to understand which features have the highest correlation, we chose the attributes *Age, MonthlyIncome* and *YearsAtCompany*. We also analysed combinations of different attributes; however, the results remained unchanged. As for the distance function, we opted for the Euclidean distance.

2.2.2 Study of the clustering parameters

We started the analysis determining the value of MinPoints, by considering the number of dimensions of our dataset (in our case 33) adding 1. After the first attempt, we tried to redouble those values, thus our MinPoints acquired the following values [34,68,136,272,544]. After this procedure, we adopted the knee method by plotting the distance to the *k-th* nearest neighbour, where k was taken from [34,68,136,272,544]. The resulting curves were used to determine the right epsilon value. In the following table we have reported respectively the k values and their Eps value.

MinPts	**Eps**
34	0.25
68	0.33
136	0.37
272	0.45
544	0.56

Figure 11: Computation of parameters

2.2.3 Characterization and interpretation of the obtained clusters

The results obtained from the different combination of the parameters reported in Figure 11, demonstrate that one single cluster is formed. High density area is located, as it is possible to see in Figure 12, in the lower part of the graph. Whereas, the orange points located in the upper area represent outliers.

The output showcases the weakness of density-based clustering, suggesting that high dimensional data and varying densities blighted the underlining analysis.

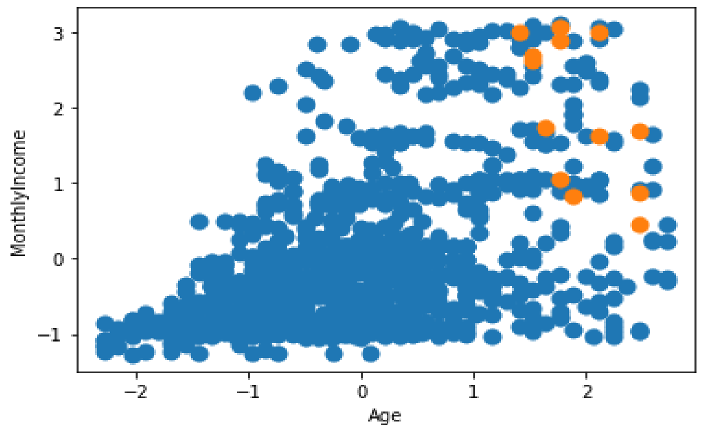


Figure 12: DBSCAN results with MinPts = 136 and Eps = 0.37.

It follows that young workers collect lower salaries compared to those who have an advanced age. With age increasing, values become more sparse, showing that as employees get older, variance between their compensation increases; whilst young employees at early stage careers have a much smaller income variance. The density area is identifying precisely this kind of population.

2.3 Hierarchical Clustering

In the following section we will discuss our procedure for clustering with hierarchical methodology.

2.2.1 Choice of attributes

For the hierarchical clustering we opted for the same attributes used for the previous clustering algorithms: *Age, MonthlyIncome* and *YearsAtCompany*

2.2.1 Affinity and linkage selection

We performed our analysis with various methods: single, complete, average and Ward's linkages and decided to compare them with both Euclidean and Manhattan distance metrics.

We first attempted to perform hierarchical clusters with numberOfCluster ∈ [2,10] for both metrics and computed their Silhouette score (Figure 13). From those plots, we notice a tendency for the silhouette to drop when the number of clusters passes from 2 to 3 for the Euclidean method and from 3 to 4 for Manhattan one.

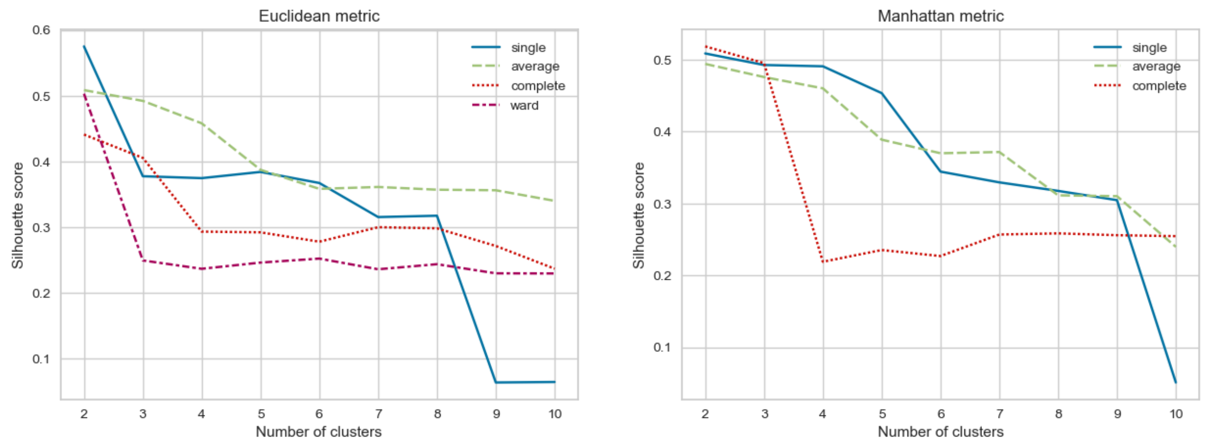


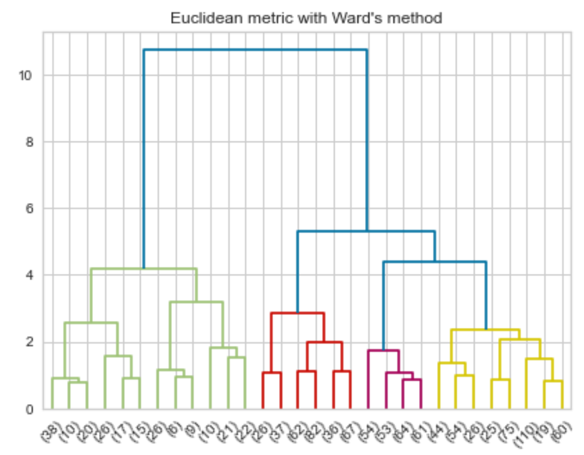
Figure 13: Silhouette scores for Euclidean and Manhattan metrics

We first tried to plot with the former numbers of clusters but realized that they yielded highly unbalanced clusters. After visually analysing the results obtained with varying numbers of clusters, we concluded that the most meaningful ones were obtained when performing clustering with 4 clusters, for both metrics. The best results were:

• Euclidean metric with Ward's method

• Manhattan metric with average method

The relative dendrograms of the above combinations are displayed in figure 14. As one can see from the color threshold, the Euclidean one was cut at 4.2 whilst the Manhattan one was cut as 0.8. The resulting clusters were fairly well equilibrated



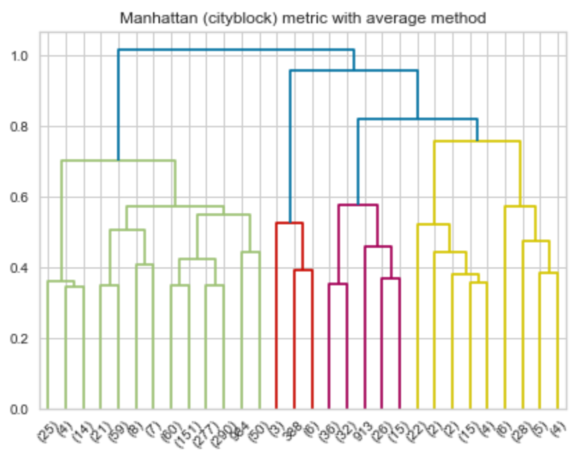


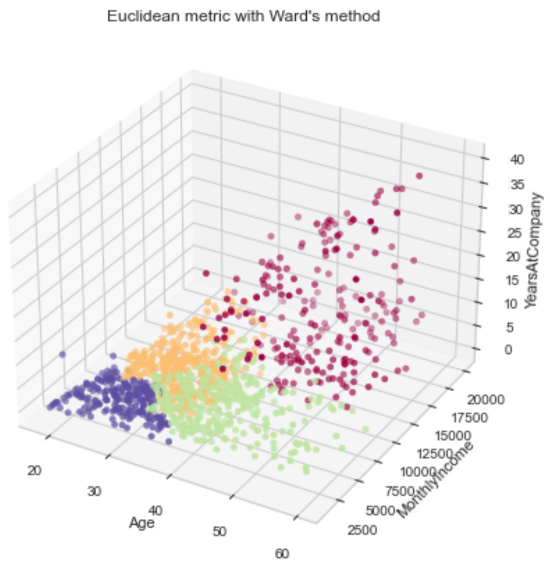
Figure 14: Dendrograms for clustering with Euclidean & Ward's and Manhattan & average method

2.2.1 Best clustering approach and comparison of the clustering obtained

We plotted the obtained hierarchical clusters on a 3D scatter plot, Figure 15. As we expected, Ward's method generated a very similar result to KMEANS clustering (as both of them rely on SSE). As, semantically speaking, Ward's results are identical to KMEANS, we refer to section 2.2.1 for the semantic explanation of Ward's based cluster.

Looking at the Manhattan based clustering, we can see a single large cluster defining the core of the dataset. Indeed, Manhattan is based on absolute value and provides a more robust, less susceptible to outliers, clustering which yields such a large, central cluster. Semantically, Manhattan's clustering is slightly different from Ward's. It identified a large cluster (in orange in the Manhattan 3D plot) that represents employees that have been in the company for just a few years (low YearsAtCompany). The blue cluster appears as employees being slightly older and having a significant higher salary. The purple cluster represents mid age employees that have stayed consistently longer in the company. Finally the green cluster represents very old employees.

Manhattan's clustering appears semantically meaningful and interestingly this result is also more valid than the Ward's when taking into account the relative Silhouette coefficient (0.48 to 0.24).



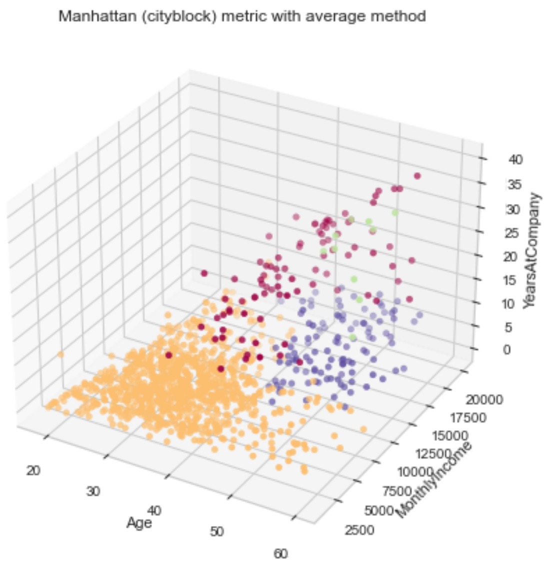


Figure 15: 3D plot of most semantically meaningful clusterings

Chapter 3 Association rules mining

In this third section, we will implement an association analysis in order to discover the connections between variables and find general rules extracted from our dataset.
In the first part we will try to extract the most frequent itemsets, trying to change the value of the support, and then to establish which type of itemset we have obtained (maximal, close, frequent). We will comment on the outputs that we have obtained.
In the second part, we will attempt to extract some association rules, in particular to find out which are the factors that lead to employees' attrition. Through the outcomes obtained, we will try to evaluate the accuracy, replacing missing values and predicting the target variables.

3.1 Attributes choices and frequent pattern extraction

Firstly, we have selected the attributes which were more interesting and explanatory for our scope: *Age*, *MonthlyIncome*, *Attrition*, *EducationField*, *Gender*, *MaritalStatus* and *JobRole*. These decisions are dictated by the choices made in the previous sections: we have continuous attributes that were discretized such as *Age* and *MonthlyIncome*, while *Attrition* was binarized with the values 0 and 1.

Regarding discretization, we have opted to use the Sturge's rule, in order to find the optimal number of classes. Thus, for the attributes *MonthlyIncome* and *Age*, we obtained 12 intervals having the same length. Furthermore, we transformed the attributes in boolean types in order to manipulate and to simplify the implementation of Apriori algorithm. To apply Apriori machine learning algorithm, we first needed to make some changes on the dataset. Using Transaction Encoder we transformed this dataset into a logical data frame. Each column represents an item and each row represents an employee.

At this point, we obtained 1176 rows and 43 columns. Firstly, we runned the Apriori algorithm with a support (hereafter called min_supp) of 10% and the maximum length of apriori items in the itemset is 3. With these parameters we finally obtained approximately 71 results.
The second iteration has been implemented with a min_sup equal to 20% and the results have fallen to 22 results. When we used closed support, the results were the same, while using maximal support the number of frequent itemsets decrease.
The iterations continued until the value of the support was lower and equal to 90%. We noticed that there is a sort of 'inverse relation', indeed if we increased the value of min_supp, the number of frequent itemsets decreased. Figure 16 illustrates this trend.

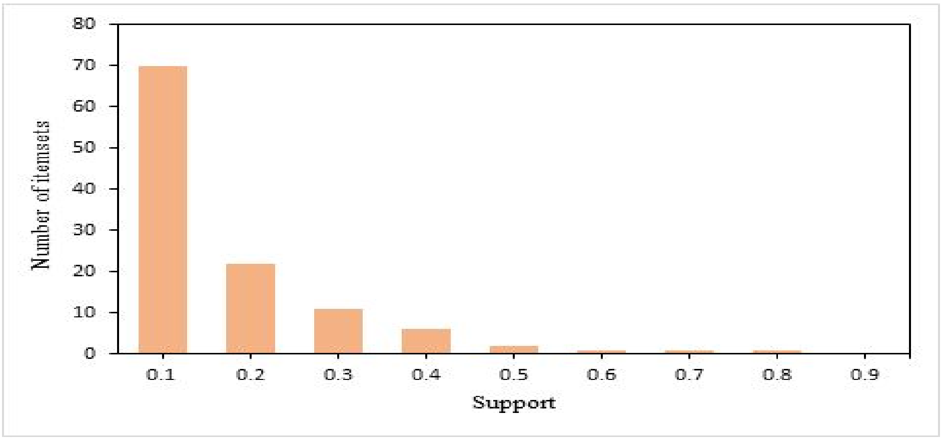


Figure 16: Itemsets with different values of support

This analysis shows that for those employees that did not have positive attrition, the frequent patterns showed that the 22% of people are married and the 20% have a background in Life Sciences. Regarding the attribute Gender, male is the type that shows up more frequently (22%) and the MonthlyIncome level is between $1730 and $3460 (12%) (Figure 17).

Itemsets	**Support**
Married, Male, Attrition:{No}	0.222
Life Sciences, Male, Attrition:{No}	0.204
Married, Female, Attrition:{No}	0.164
Life Sciences, Married, Attrition:{No}	0.162
Medical, Male, Attrition:{No}	0.144
Life Sciences, Female, Attrition:{No}	0.128
Income:[1730-3460], Male, Attrition	0.119
Female, Medical, Attrition:{No}	0.112
Life Sciences, Single, Attrition:{No}	0.109
Male, Divorced, Attrition:{No}	0.105

Figure 17: Itemsets of different types with support between 10% and 90%

3.2 Association rules extraction, evaluation of accuracy, replacing missing values and prediction of targeting variables

In this section we will discuss the most significant rules extrapolated from the Apriori algorithm. We have established to extract rules with a min_supp equal to 0.1, a length of maximum three attributes and confidence lower or equal than 85%. The histogram in Figure 18 illustrates how the number of rules varies by varying the confidence value (the confidence interval selected was between [85% - 10%]). We found optimal results with support equal to 20% and confidence 85%. The association rules found with those optimal parameters are shown in Figure 19.

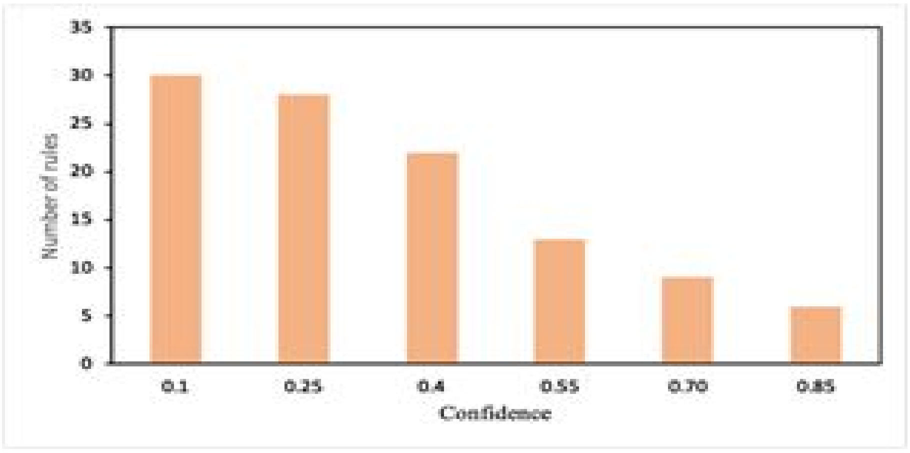


Figure 18: Rules with different values of confidence

Antecedents	Consequents	Confidence	Lift
Male, Married	Attrition:{No}	0.885	1.057
Married	Attrition:{No}	0.876	1.047
Medical	Attrition:{No}	0.862	1.030
Income:{$1730 - $3460}	Attrition:{No}	0.853	1.019
Male, Life Sciences	Attrition:{No}	0.851	1.017
Life Sciences	Attrition:{No}	0.850	1.016

Figure 19: Association rules with the optimal parameters

The Figure above illustrates that employees with negative *Attrition* are married males that gain a monthly income ranging between [$1730-$3460], with a medical education background or in life sciences.

3.3 Evaluation of accuracy, replacing missing values and prediction of targeting variables

Concerning the replacement of missing values, we decided to replace the missing elements of the attributes Age and Gender, because both presented some lacking in the dataset. Instead, for the prediction of the target variable, for those who have *Attrition* equal to no, we replaced using the rules found above with the highest values of confidence and lift (*Married - Male - Life Sciences - Income* between [$1730-$3460]).

On the contrary, in case of positive answers to *Attrition* in the workplace, we used a min_support equal to 0.01 because for higher values we did not obtain any rules and patterns. We therefore found and replaced those who have no partner (singles) with an occupation as a Sales Representative. Once more, *Life Sciences* was the educational field where the results obtained were more significant.

Chapter 4 Classification

In this section we will explain our procedure for building a classification model. The model is designed to solve a binary classification problem, it will use a set of the dataset's attributes to predict whether an employee has a positive attrition. We will first proceed with a classification technique known as decision tree classifier and then compare the results obtained with KNN classifier (kth nearest neighbor).

4.1 Decision tree classifier

In the following section we will discuss our procedure for solving a predictive classification problem using the decision tree classifier algorithm.

4.2.1 Preprocessing, choice of attribute set and dataset split

Before starting the induction process, we firstly proceeded to manually encode (for preserving the order) the only qualitative ordinal variable that still had a non-numeric representation: *BusinessTravel*.
We then encoded with OneHotEncoder (OHE) *EducationField*, *JobRole* and *MaritalStatus* as they were non-binary qualitative nominal variables with numerical encoding. The rationale behind this decision was that the classifier might consider them as ordinal or numeric variables and alter its split test condition.
As a consequence of OHE, our dataset now consisted of 38 variables.

Relating to the dataset split, we decided to split our current dataset for obtaining a validation set which we would use to tune the classifier's hyper-parameters. The dataset was accordingly split into the attribute set 'X' and the target variable 'y' containing the binary class label *Attrition*. Those sets were then randomly splitted into train and validation subsets using stratification so as to better balance our samples.

4.2.1 Hyper-parameter tuning

As one could expect, the first run of our algorithm, which was instantiated with no early stopping constraints, developed a full decision tree and was naturally strongly overfitted having 100% accuracy on the train set and only 75% on the validation set.
A hyper-parameter tuning for pre-pruning our model was necessary.
The main parameter of the decision tree classifier we decided to tweak were *max_depth, min_samples_leaf* and *min_samples_split.*
We structured the parameter tuning in two parts:

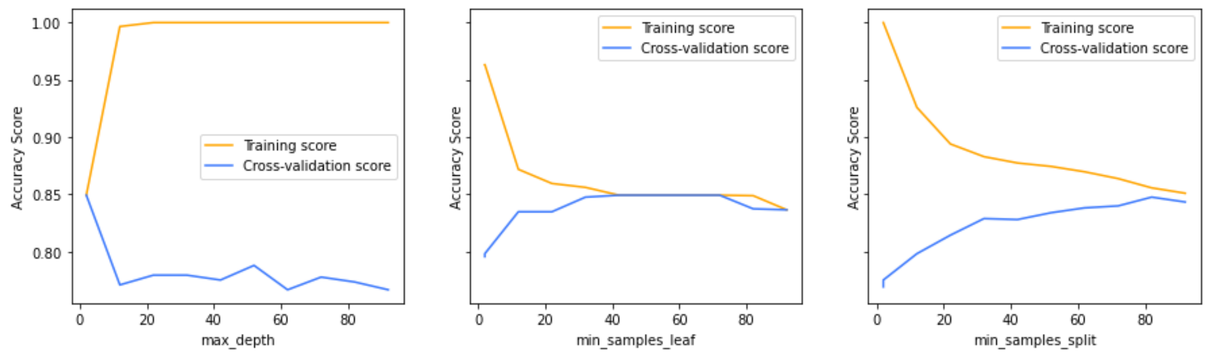


Figure 20: accuracy score for various values of *max_depth, min_samples_leaf* and *min_samples_split*

In the first part, we plotted the validation curve obtained for each main parameter over a predetermined range of values.
From figure 20 we can see a rather poor performance of our classifier for any value of the parameters (although it should be noted that when plotting the validation curve, the other parameters are left to their standard values), which already warns us that our model might perform poorly. Indeed we can see that for any value any parameter, cross-validation accuracy does not surpass 85%.
Given the poor performance of the classifier for almost any values of the hyper parameters, we proceeded with a supervised feature selection method to see whether reducing the number of input variables would improve the model's performance. We started with a filter based feature selection to select the most appropriate variables. We used univariate statistical measures to evaluate the dependency of each variable with the target variable *Attrition* (using ANOVA for quantitative and Chi-squared for qualitative attributes). Figure 21 summarizes the results.

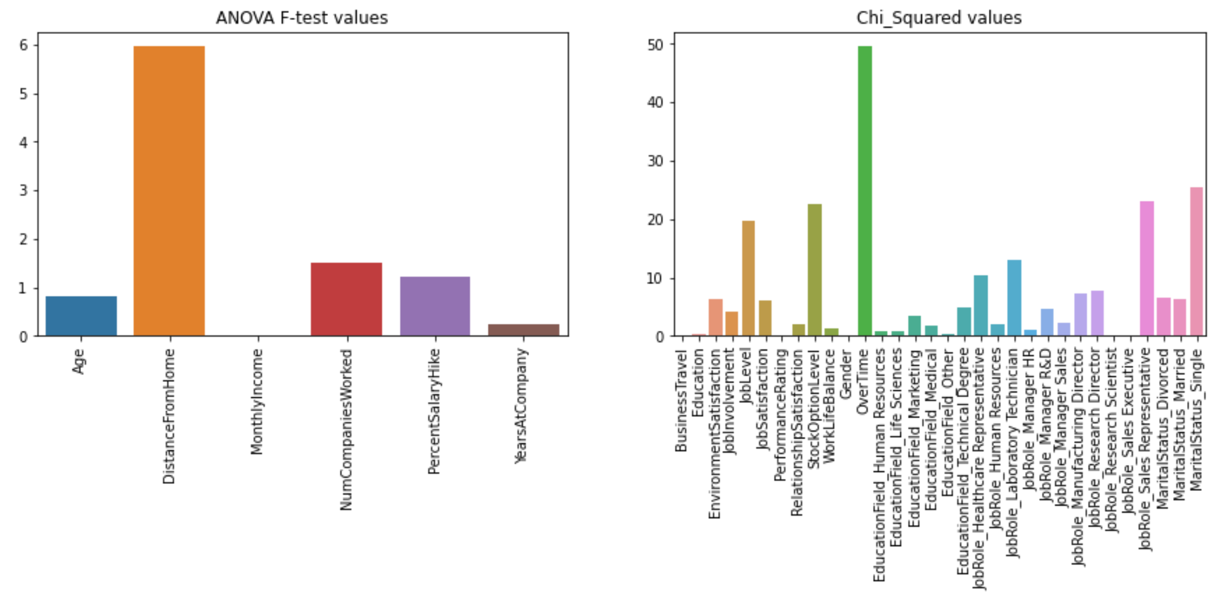


Figure 21: F-test scores for ANOVA and Chi-Squared

In both cases (ANOVA and Chi-Squared), the higher the score, the higher the chances of the variable being dependent on the target variable. As we can see from Figure 21, some variables have a particularly high dependency on the target variable. Notably *DistanceFromHome* and *OverTime*.
Using filter based feature selection allowed us to quickly make a first sub-selection of attributes for our classifier. By setting a threshold on ANOVA higher than 1 and Chi-Squared value higher than 10, we obtained a sub-selection of 10 attributes.

We then tried training our model on the new subset but no significant improvement was seen. We therefore decided to proceed with further attribute sub-selection. Our choice was to use a wrapper based feature selection algorithm. Being left unsatisfied with the results obtained by Sklearn' s RFE, we decided to create our own wrapper algorithm. We named our algorithm TheBeast and a simplified version of the code can be found in the appendix.
Our *TheBeast* had the following procedure:
Consider n the number of attributes of a dataset and k the number of selected attributes at once (ranging from 1 to n):
Loop over k and proceed with combinations identification. Loop again on all possible combinations for a given k and proceed with a light GridSearch (few parameters just for enabling comparison).
For every k, save top performing combinations.

Once iterations over all possible *k* has finished, the algorithm has a collection of the best performing combinations of all possible subsets of the initial dataset (with a cross-validated score by GridSearch). The algorithm then takes the top 5 combinations (ranked by score, for any value of *k*) of this collection and proceeds with a much deeper GridSearch (many parameters to identify best possible result for every combination). Finally, TheBeast returns the single best performing sub-set combination along with its parameter configuration and cross validation score. More information about TheBeast can be found in the appendix.

4.2.1 Training Results

The following is the result of our custom wrapper algorithm.
The best variable mix identified was made of 5 variables (1 numeric, 3 binary, 1 ordinal):
NumCompaniesWorked, OverTime, MaritalStatus_Single, JobRole_Laboratory Technician, JobLevel

The best parameter values for the classifier were: *max_depht = 6, min_samples_leaf* = 13 and *min_samples_split* = 5
Cross-validation score: 86.31 % (+/- 0.04)
F1-score: 65.61 % (+/- 0.14)

Features importances also indicated that *OverTime* is the most important feature for the classifier, with an importance of 0.34, closely followed by *JobLevel* with an importance of 0.30. Note that *DistanceFromHome* (the most dependent quantitative feature) was not selected by our algorithm. We tried running a separate classifier including this variable but the results were indeed lower, confirming the functioning of our TheBeast.

The resulting decision tree is shown graphically in the appendix. We reported only the first levels as after that only 'no Attrition' conditions were left.

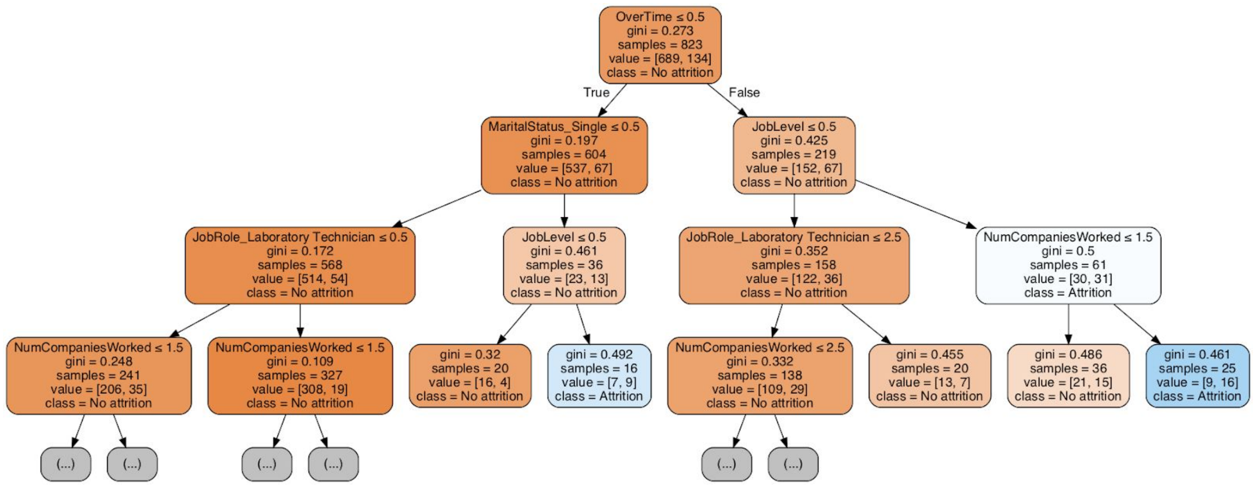


Figure 22: the Decision Tree Classifier obtained

As we can see the tree reflects our low accuracy score. The only two 'Attrition' terminal nodes have very high gini coefficient scores, disclosing almost identical repartition and thus making our model not particularly accurate.

4.2 K-nearest neighbors

In the following section we will discuss our procedure using the K-nearest neighbors for solving a predictive classification problem.

4.2.1 Preprocessing, choice of attribute set and dataset split

We tried different ways of preprocessing the data frame. At first, since KNN is an algorithm based on the distance, we removed all the categorical attributes that did not have an ordinal nature and could not be encoded manually. We also tried to include them by encoding their value with OneHotEncoder (OHE) but the test using this datas gave us worse results.

Another test we did was to normalize the data, since KNN measures the distance between pairs of samples and can be influenced by the measurement units. But even in this case the test produced worse results.

2.3.1 Choosing the best K

In order to select the right value for K, we run the KNN algorithm several times with different values of K and choose the K that has the best Accuracy and F1-score.
The result we obtained are the following:

K	Accuracy	F1-score
2	0.8204 (+/- 0.01)	0.4685 (+/- 0.01)
3	0.8008 (+/- 0.02)	0.5135 (+/- 0.01)
4	0.8281 (+/- 0.01)	0.4762 (+/- 0.03)
5	0.8281 (+/- 0.01)	0.4925 (+/- 0.01)
6	0.8340 (+/- 0.01)	0.4788 (+/- 0.03)
7	0.8272 (+/- 0.01)	0.4804 (+/- 0.02)
8	0.8323 (+/- 0.00)	0.4641 (+/- 0.02)
9	0.8272 (+/- 0.01)	0.4624 (+/- 0.02)

Figure 23: Accuracy and F1-score performance for different value of K

As we decrease the value of K to 1, our predictions become less stable. Inversely, as we increase the value of K, our predictions become more stable and more likely to make more accurate predictions until we push the value of K too far and the number of errors increase. From the figure 23 we can see that the best value for K is between 4 and 7.

2.3.2 Choosing the best subset of Attributes

Similarly to the decision tree classifier the results, using the classifier with all the attributes, did not provide acceptable results. We therefore used a slightly modified version of TheBeast, to find the best subset of attributes that maximizes accuracy and the F1-score of the cross validation.

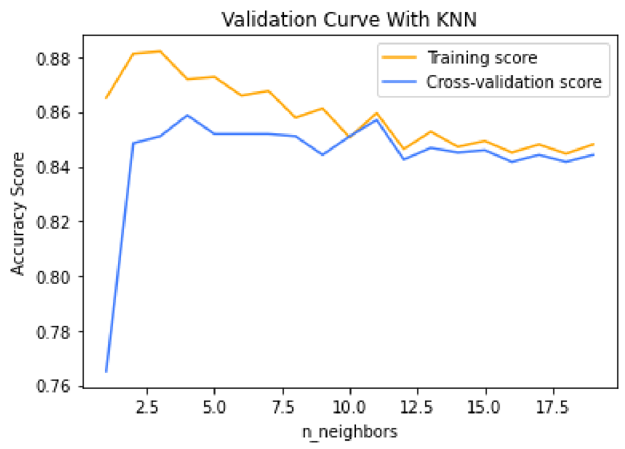


Figure 24: Accuracy score for various values of k changes

The script identified the subset *['EnvironmentSatisfaction', 'JobLevel', 'OverTime', 'StockOptionLevel', 'WorkLifeBalance']*, that passed to the classifier together with the value of k equal to 4 obtains an accuracy of 0.8587 (+/- 0.01) and an f1-score of 0.6358 (+/- 0.01).

4.3 Test results and classifier comparison

After developing our model on the training dataset provided, we eventually tested our model with the test dataset provided (Test_HR_Employee_Attrition.csv). The following results were the result of a single shot prediction for the 294 records of the test set:

Decision tree classifier:
Test accuracy: 83.34 %; Test F1-score: 30.34%
KNN classifier:
Test accuracy: 84.01 %; Test F1-score: 22.95%

Overall our classification models performed quite poorly. Considering that a basic algorithm predicting always *Attrition* = No would have scored around 84% on the test dataset, we can say that, accuracy wise, our models are not better than random guessing. However, accuracy is not particularly meaningful due to the imbalanced dataset. The F1-scores and the ROC curve indicate that although the accuracy of our models was the same than random guessing, they behaved slightly better than a random model. Indeed, a classifier always predicting "No Attrition" would have had a recall and precision of 0 (so an F1-score of 0 as well). In Figure 25,we can see that the plotted Roc Curve for both classifiers is slightly higher than the black diagonal indicating the performance of a random guessing classifier.

Overall our classifiers have a similar performance. Looking at their confusion matrices we saw and from the Roc Curve, we understand that the models are not able to well distinguish between classes (ie. low class separation capacity).

As for comparing the classifier between them, we can see that the decision tree classifier, as indicated by the higher F1-score, scored a slightly higher area under Roc Curve meaning that it was slightly better than the KNN classifier. Indeed we can see that for higher thresholds, the decision tree's recall increases much faster than KNN's. The same result is visible on the classifier F1-scores. Therefore, we concluded that the decision tree achieved a (slightly) better performance.

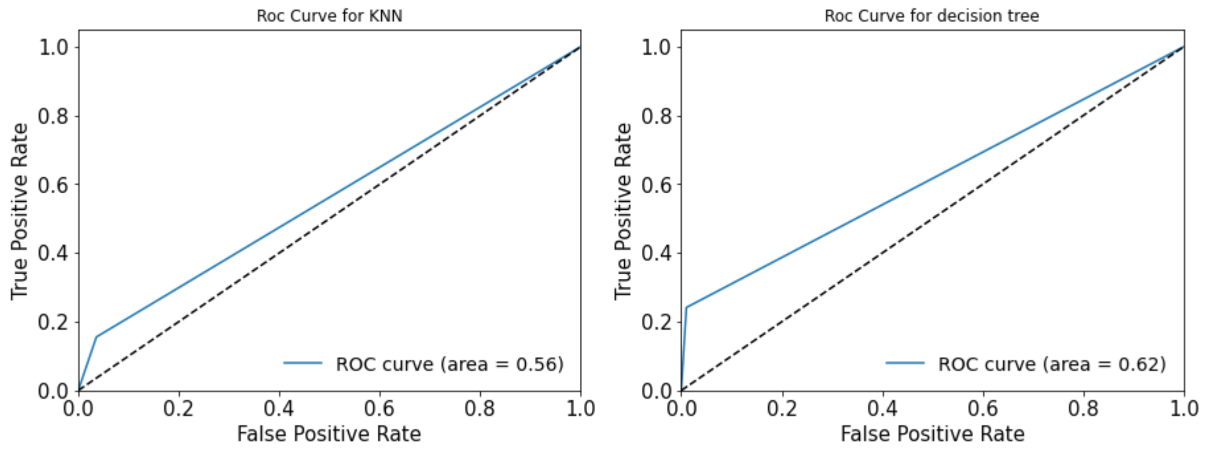


Figure 25: Roc Curve comparison between KNN and decision tree classification

Conclusion

Using the "IBM HR Analytics Employee Attrition & Performance" dataset, we tried different data mining techniques in order to obtain actionable insights regarding employee attrition.

We started with context and data understanding, providing the reader a clear picture of the data we used. From data cleaning to transformation and description we explained our procedure for understanding and preprocessing the available variables. We visually showed the reader that numerous variables had a substantial amount of inconsistent records and had to be removed from our further analysis. We guided the reader through our variable imputation and data cleaning procedure. We furthermore transformed some features to correct their asymmetries and render them more usable for further analysis.

In the *Clustering* section we tried to find patterns in the data that would identify the presence of natural clusters. We tried different clustering techniques studies and visually inspected the clusters created by the algorithms. Some clusters, such as K-Means and hierarchical with Ward's method yielded interesting and semantically meaningful clusters and thanks to clustering we learned about different potential types of employees. We further analyzed the statistical impact of clusters classes on the employees' attrition rate but saw that they were insignificant.

We then proceeded with association rules analysis. We successfully looked for patterns that would clarify and respond to our initial questions, specifying which are the attributes that lead an employee to engage in voluntary attrition in the workplace.

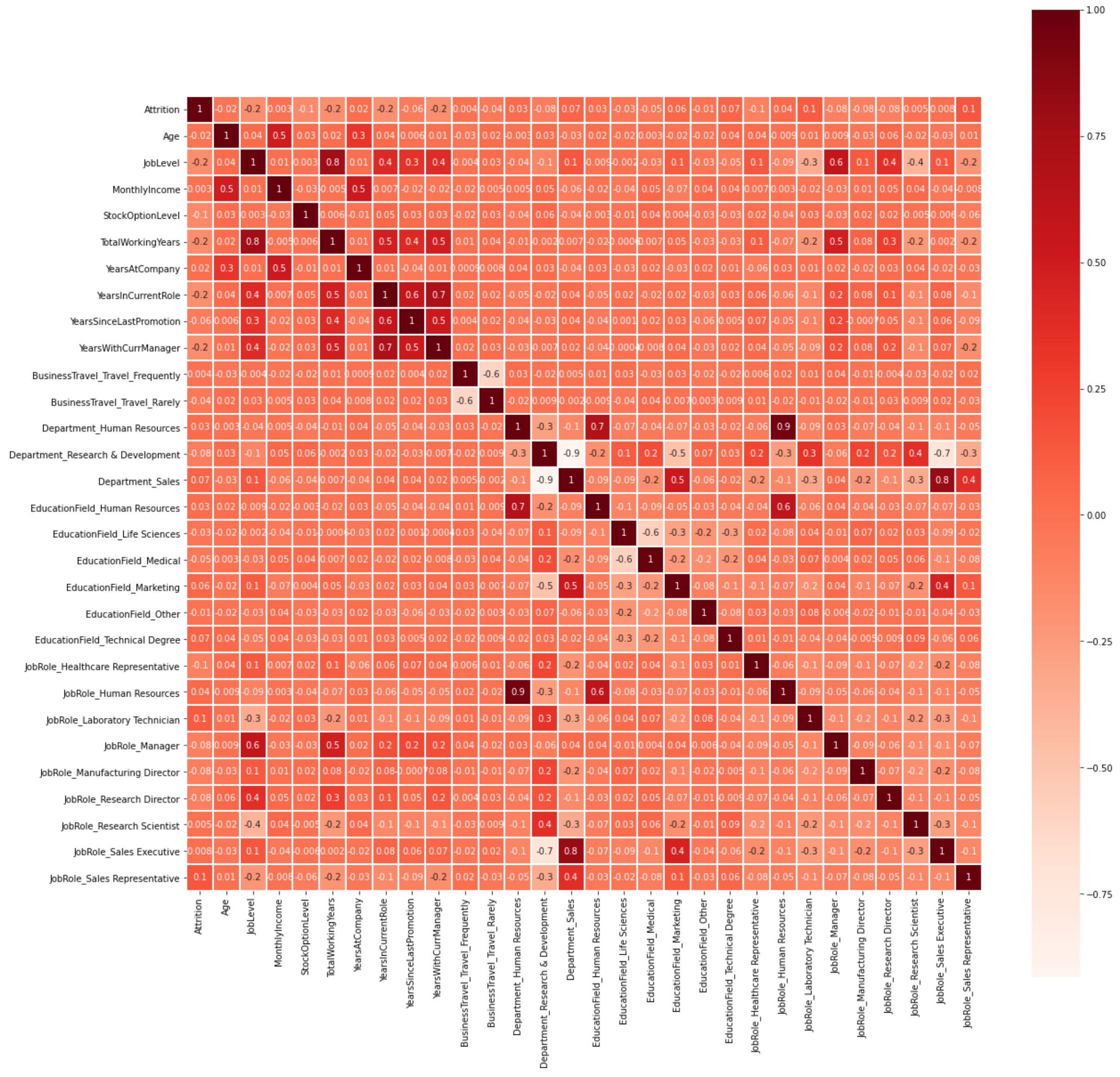
Finally, in the last part we tried to predict the value of the target variable, namely *Attrition*. We wrote a custom script to find the best possible subset of attributes and the best hyper-parameter values. We successfully trained our models and fine tune them using the appropriate techniques. We eventually proceeded with induction to predict on a test dataset whether a list of employees would pursue voluntary attrition. The performances of our classifiers were far from optimal and thus indicated that we probably missed an essential data preprocessing procedure. Nevertheless, we showed that should the reader use one of our classifiers to check which of his employees will have a positive attrition, he would statistically obtain better results than by always predicting negative attrition; which is, we think, a positive first milestone.

Thank you for reading this far,

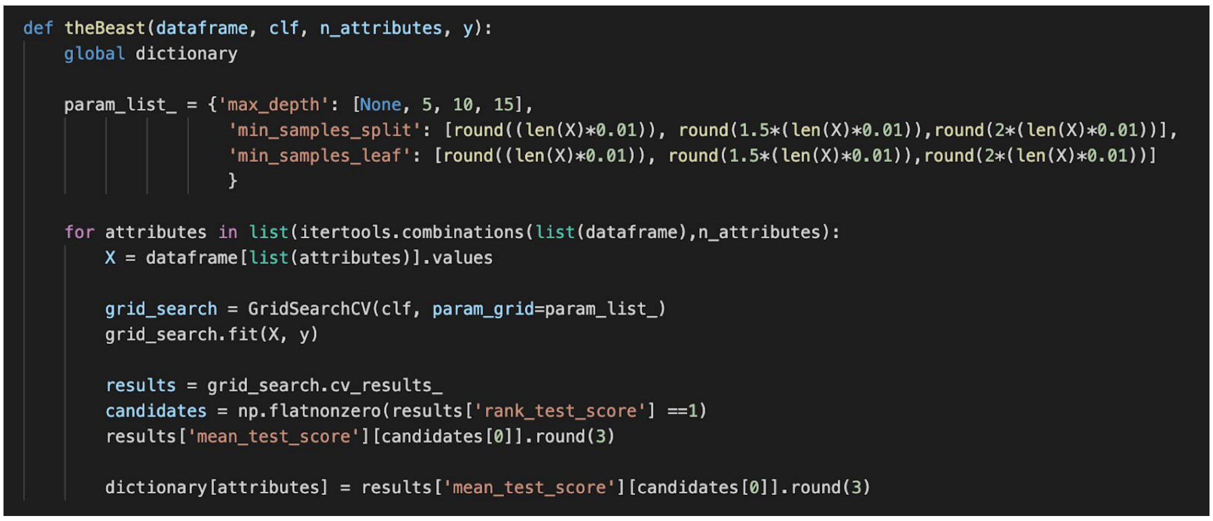
Blerta, Claudio, Giovanni, Agnese

Appendix

A. **Correlation Map** with all the attributes converted in numeric attributes.



B. The Beast relies on a double multi process concurrency for parallelizing the execution of the loop on different cores and achieving notable performance improvement. At first sight, it might indeed seem tremendously computationally expensive. However, it only took around 4 minutes of running time to find the best possible subset on our 10 attributes dataset.
A simplified version of the algorithm is portrayed below. The full version we used for our finding the best variables and parameters mix for our classification model can be found on the following repository: https://github.com/giogioia/Classification_Beast



```
def theBeast(dataframe, clf, n_attributes, y):
    global dictionary

    param_list_ = {'max_depth': [None, 5, 10, 15],
                   'min_samples_split': [round((len(X)*0.01)), round(1.5*(len(X)*0.01)),round(2*(len(X)*0.01))],
                   'min_samples_leaf': [round((len(X)*0.01)), round(1.5*(len(X)*0.01)),round(2*(len(X)*0.01))]
                   }

    for attributes in list(itertools.combinations(list(dataframe),n_attributes):
        X = dataframe[list(attributes)].values

        grid_search = GridSearchCV(clf, param_grid=param_list_)
        grid_search.fit(X, y)

        results = grid_search.cv_results_
        candidates = np.flatnonzero(results['rank_test_score'] ==1)
        results['mean_test_score'][candidates[0]].round(3)

        dictionary[attributes] = results['mean_test_score'][candidates[0]].round(3)
```

Screenshot of TheBeast simplified version