Università di Pisa
Department of INFORMATICA
Master of Data Science and Business Informatics

Course: Data Mining- Foundations

PROJECT REPORT
IBM HR Analytics Employee Attrition and Performance

Submitted by:
Elisa Pashku (628386)
Luca Palla (533605)
Martina Sustrico (533252)
Matteo Garro (620025)

Academic Year 2020/2021

Table of contents

Data Understanding

This report examines the *HR Analytics Employee Attrition & Performance* dataset created by IBM. The real-world phenomenon the dataset tries to model is attrition. The money, time and effort invested in training new employees leads to a massive overall loss to the firm when an 'old' employee leaves. Understanding and recognizing what factors are associated with employee attrition will allow companies to limit this from happening and may even increase employee productivity. These predictive insights give managers the opportunity to take corrective steps to build and preserve their business' success.

The dataset is divided into a train and test set. To develop a thorough analysis, both datasets are merged into one. Therefore, there are *1470 records* (observations) and *33 features* in total.
Python 3.3 is used for analytics and model fitting. The IDE used is *Jupyter*.

1.1 Data Semantics

The independent variables (attributes, features) are *32* and the dependent variable is *Attrition* with possible outcomes *Yes* and *No*. Table 1 presents the name, description, type and domain of 8 attributes chosen randomly.

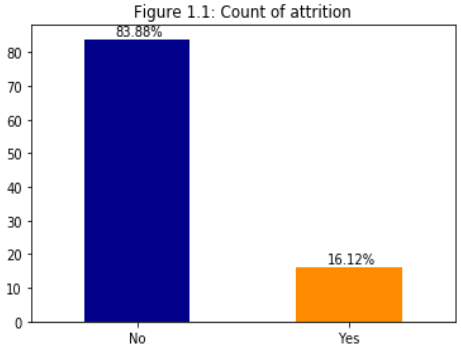
Table 1.1 *Description of some attributes*

Name	Description	Type	Domain
Age	Age of employee	Numerical - Discrete	A > 18
Attrition	Whether the employee has attrited or not	Categorical	{yes, no}
BusinessTravel	How frequently the employee has traveled	Categorical	{Travel_Rarely, Travel_Frequently, Non-Travel}
MonthlyIncome	Employees' income for month	Numerical - Discrete	[1099, 19999]
YearsAtCompany	Number of years that each employee spent in company	Numerical - Discrete	[0, 40]
DistanceFromHome	How far away from home is the work	Numerical - Discrete	[1, 29]
EnvironmentSatisfaction	Rating of job's environment satisfaction	Categorical	{1,2,3,4}
Gender	Gender of employee	Categorical	{Male, Female}

1.2 Distribution of the variables and statistics

This section presents the distribution of some of the variables with the help of histograms and count plots and the statistics of the same variables.

Firstly, the chart (*Fig 1.1*) shows the count of the distinct values of the target variable. Out of the total of 1470 observations, ***16%*** have left the job compared to ***84%*** that have stayed. This is a case of *data imbalance*, with a majorly overrepresented class.



Observing *figure 1.2*, there appear a lot of employees that are 'newbies' in the company (*Job Level* 1-2), and examining the plot of *Business Travel* (1.2.(f)) it is assumed that a relation exists between a low *JobLevel* and the category "*Travel_Rarely*".

Figure 1.3 shows that Age has a close to normal distribution, instead *DistanceFromHome*, *MonthlyIncome* and *TotalWorkingYears* have a left-skewed distribution.

In the table 1.2 are represented some statistics about the attrition incidence on *JobLevel*, *OverTime* and *MaritalStatus.*

Table 1.2 *Attrition for Job Level, Marital Status and OverTime*

JobLevel	1	2	3	4	5
Attrition					
No	32.44	39.09	15.09	8.19	5.19
Yes	60.34	21.94	13.50	2.11	2.11

MaritalStatus	Divorced	Married	Single
Attrition			
No	23.84	47.77	28.39
Yes	13.92	35.44	50.63

OverTime	No	Yes
Attrition		
No	76.56	23.44
Yes	46.41	53.59

The float numbers in the cells are the percentage of attrition (Yes/No) considering the different category in the columns. It is easy to see that for a *JobLevel* equal to 1 (the lower) there is a higher percentage of 'Yes' *Attrition* (60.34%); at the same, employees that work more time than necessary are more likely to quit their job (as much as 53.6%). Single employees have the higher attrition of the *MaritalStatus'* set, on the other hand Married and Divorced people are more likely to not leave the job.

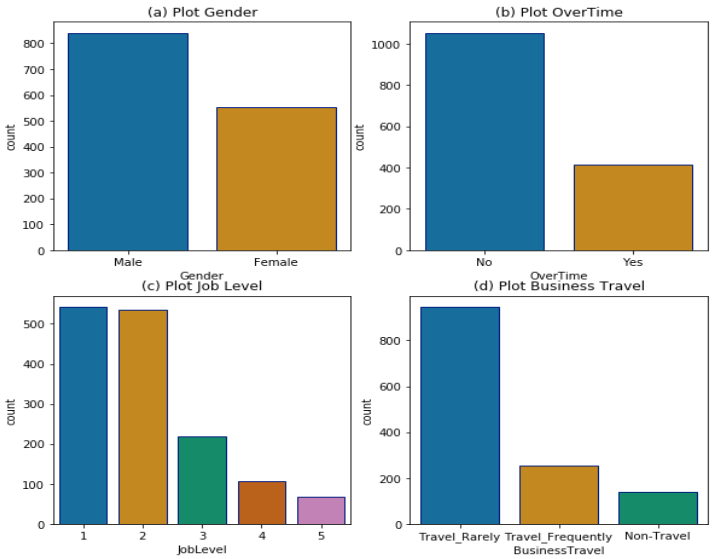


Figure 1.2 *Countplot of categorical attributes*

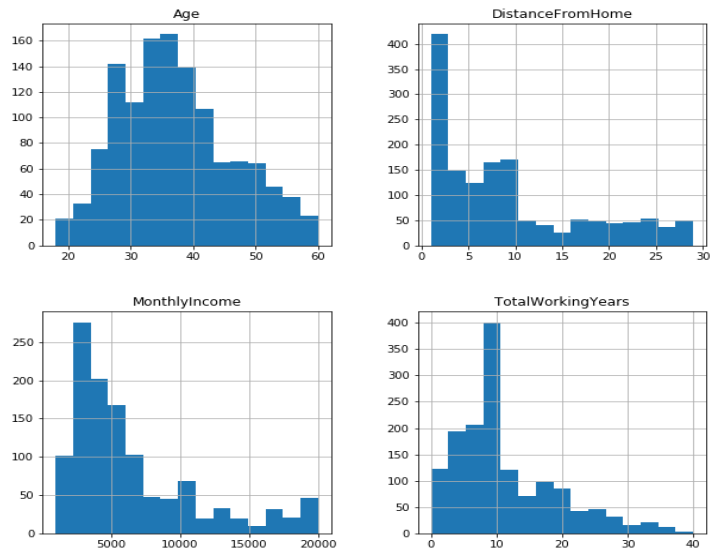


Figure 1.3 *Countplot of numerical attributes*

1.3 Assessing data quality

This part evaluates the quality of data. It investigates and handles missing values and outliers.
Firstly, it has been decided to investigate the dataset for the *none* values, and to simplify the next work those were replaced using a specific method (see chap. 1.3.2). Finally, Z-score method helped identifying outliers in the data.

1.3.1 Missing values

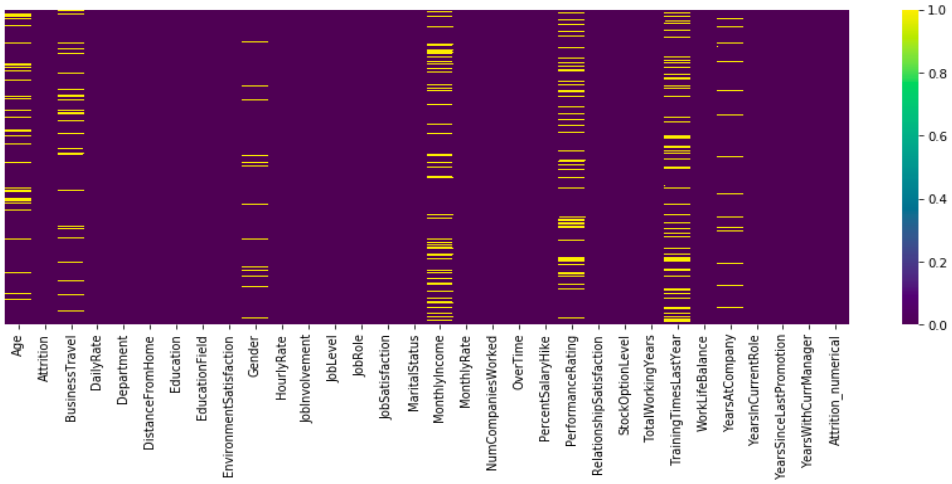
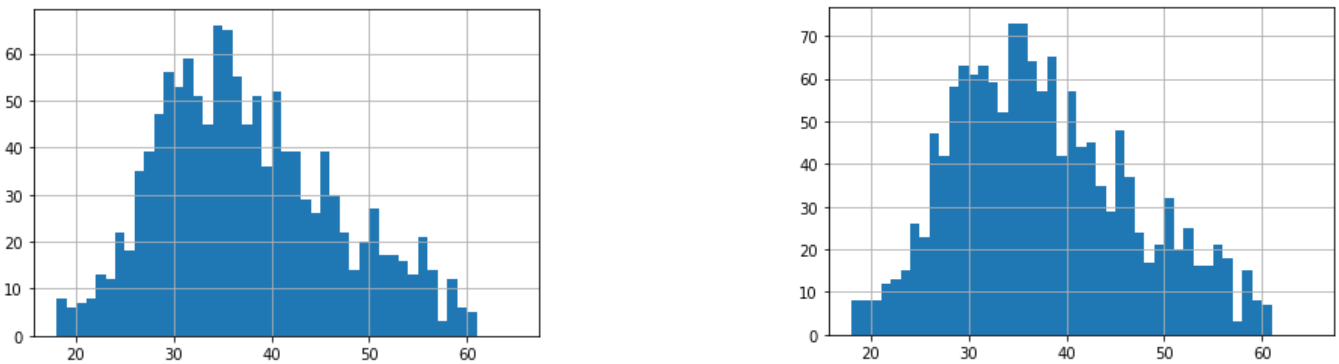


Figure 1.4 *Heatmap of missing values*

The heatmap **(**Figure 1.4) visualizes that *Age*, *BusinessTravel*, *Gender*, *MonthlyIncome*, *PerformanceRating*, *TrainingTimeLastYear*, *YearsAtCompany* are the features with missing values.
Going into detail, *Age* has got 212 observations, *BusinessTravel* has 131, *Gender* has 75, *MonthlyIncome* has *280*, *PerformanceRating* has 172, *TrainingTimeLastYear* has 292, *YearsAtCompany* has 74.

1.3.2 Filling the missing values

Considering that the dataset is *unbalanced*, using the *mode* or *median* for the missing values, would deepen this unbalance. To make this process random but also not to alter the original distribution in a significant way which could alter future results, the used approach considers the original probability of distribution of the variables. As an example, *figure 1.5* shows that the frequency of the attribute *Age* after replacing of missing values (b) is proportional to the old one (a).



(a) Before replacing *(b) After replacing*

Figure 1.5 *Frequency of Age attribute*

1.3.3 Outliers

To identify anomalies within the dataset, Z-score method was implemented. Significant anomalies were found in *YearsAtCompany* and *YearsInCurrentRole.*

After calculating the Z-scores of the numerical attributes and restricting an absolute threshold of 3, there appeared 35 outliers in the dataset. Outliers have been handled by dropping them since it would not affect future analysis in a meaningful way. Figure 1.6 and figure 1.7 show the standardized distribution of *YearsAtCompany* and *YearsInCurrentRole*; the values with Z-score greater than 3 are considered outliers.

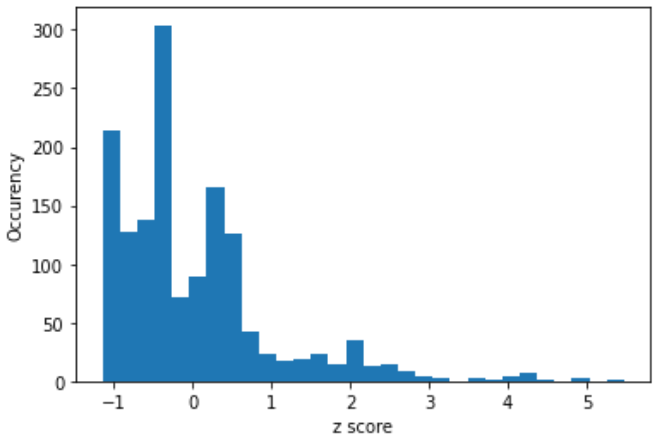


Figure 1.6 *Standardized distribution Years at Company*

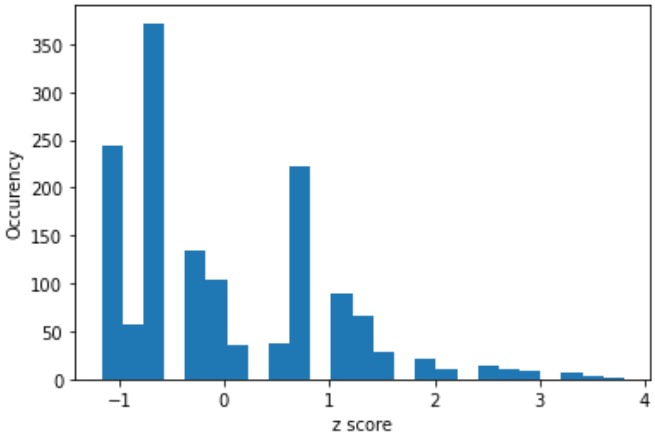


Figure 1.7 *Standardized distribution Years in current role*

1.4 Variables Transformation

First, the dependent variable *(Attrition)* is converted into a numerical variable by applying **0** if the response is 'No' and **1** otherwise.

Attrition	Attrition numerical
Yes	1
No	0

Table 1.3 *Attrition converted into numerical*

Secondly, *Over18* and *StandardHours* are columns which do not provide much new information:

- The *Over18* column is dropped because the minimum employee' age is **18** (all the records have 'yes').
- The *StandardHours* is dropped as **80** is the only value it contains.

In addition, given the nature of *DailyRate, HourlyRate and MonthlyRate* it is safe to assume that from an economic point of view they should be correlated between each other. In the dataset appears no correlation even though they should all be deducted from *MonthlyIncome*. A possible solution would have been to find the relation between them but in this case, they would provide no further information but just a new point of view of an already existing variable (*MonthlyIncome*) therefore, they are dropped.

Going into deep of the data has been discovered some inconsistency, about the *Years in Company* and *Total Working Years* compared to employees' age: some young employees work from more time than we can assume.

Then, in view of the data management with accurate methods in the following chapters, categorical attributes have been transformed into numeric.

1.5 Pairwise correlations and elimination of redundant variables

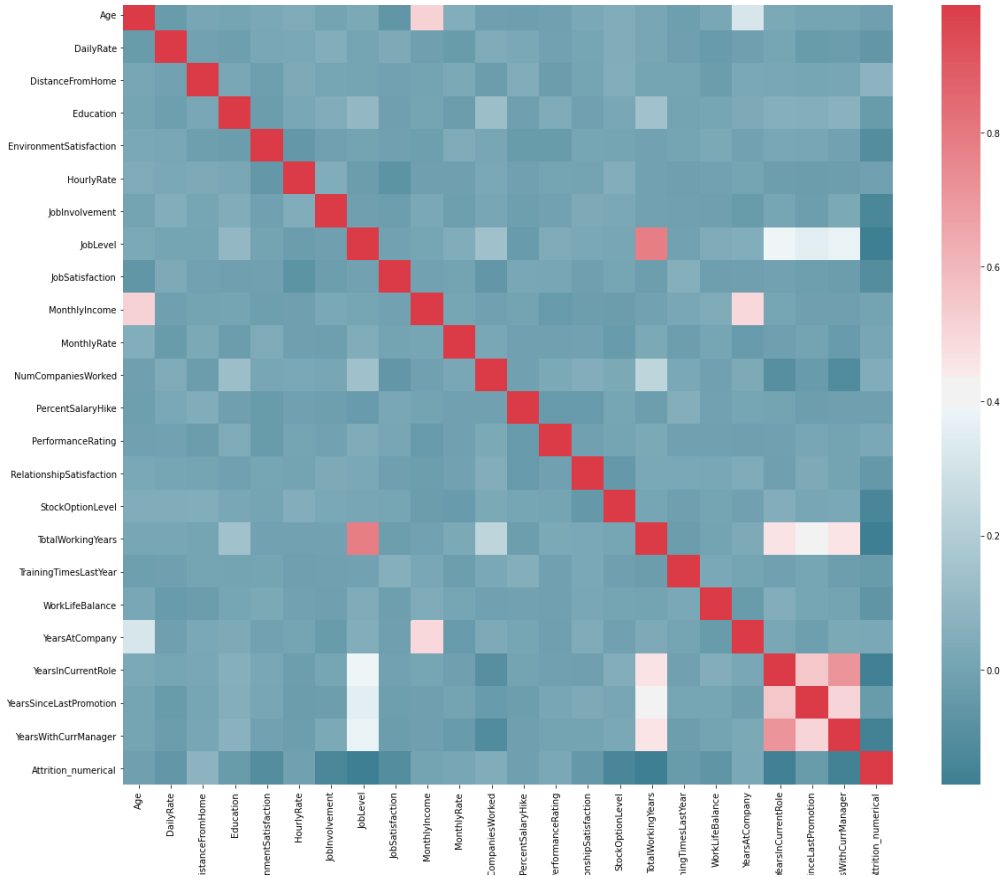


Figure 1.8 *Heatmap of correlation matrix*

The heatmap (Figure 1.8) shows the correlation matrix, where the red square corresponds to a high correlation and the blue one to a low correlation. The most highly correlated pairs are *YearsInCurrRole* and *YearsWithCurrManager* (0.71), *JobLevel* and *TotalWorkingYears* (0.78), *YearsInCurrRole* and *YearsSinceLastPromotion* (0.55). Since these pairs are highly correlated, to simplify the model, *YearsWithCurrManager*, *TotalWorkingYears* and *YearsSinceLastPromotion* columns are dropped.

As an example, considering working years, it is common to assume that an employee that works a lot of years, gains more experience, and gets a higher *JobLevel*.

Same reasoning can be applied to *YearsInCurrRole* and *YearsSinceLastPromotion* and *YearsInCurrRole* and *YearsWithCurrManager*

Clustering

In the following section, different methods are used to perform the clustering. The algorithms applied are K-means, density-based (DBScan) and hierarchical clustering.

In the IBM Attrition dataset, the main goal is to predict the variable ***"Attrition"*** and since clustering is an unsupervised task, "Attrition" is removed from the data frame. Clustering algorithms require binary or numerical attributes; thus, the categorical attributes are saved in a separate data frame and dropped, so it is possible to continue the analysis with only continuous values.

To increase the performance of the clustering algorithms, the dataset's attributes are standardized using StandardScaler.

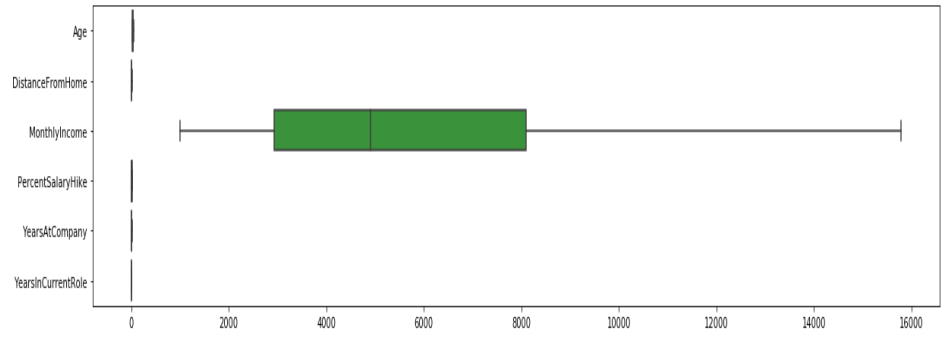


Figure 2.1 *Boxplot of variables before standardization*

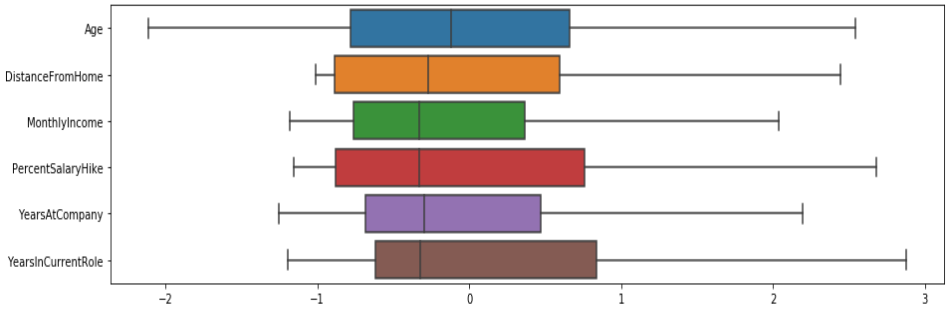


Figure 2.2 *Boxplot of variables after standardization*

In figure 2.1, *MonthlyIncome* is many orders of magnitude higher than the other variables, to make the columns comparable their values have been standardized. The result of the standardization can be seen form figure 2.2, where attributes have mean near the zero and standard deviation near to 1.

2.1 Clustering Analysis by K-means

The K-means algorithm clusters data by trying to separate samples in **k** groups of equal variances minimizing the within-cluster sum-of-squares (inertia).

2.1.1 Choice of attributes and distance function

In K-means analysis has been decided to perform the algorithm with the following attributes: *MonthlyIncome*, *Age*, *YearsInCurrentRole*, *YearsAtCompany*, *PercentSalaryHike*, *DistanceFromHome*. Regarding the implementation, the distance function considered is the Euclidean.

2.1.2 Identification of the best value of k

K-means algorithm requires a specified number of clusters **k**. The choice of k is important because with values that are too low or too high, little significant results can be obtained. The value of k is found by taking into consideration the trend of SSE and the silhouette as the k.

The focus is put on the 'knee of the curve' in the plot (figure 2.3) -that is the interval between 5 and 10-. According to the silhouette metric it is possible to confirm that **6** is the value

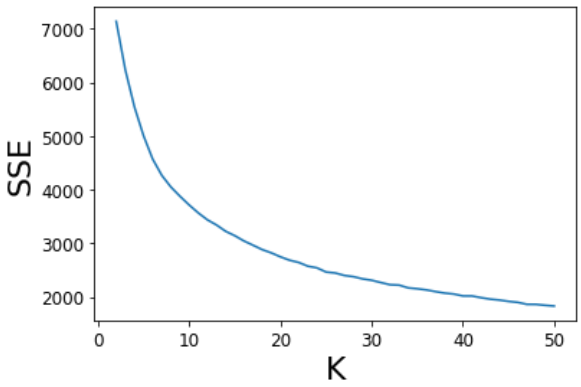


Figure 2.3 *Sum of Squares Error (SSE)*

which maximize Silhouette score (0.1853) and where adding another cluster does not give much better modeling of the data.

2.1.3 Characterization of the obtained clusters

The figure on the right (figure 2.5) shows the graphical results of k-means clustering with three sample attributes taken in examination.

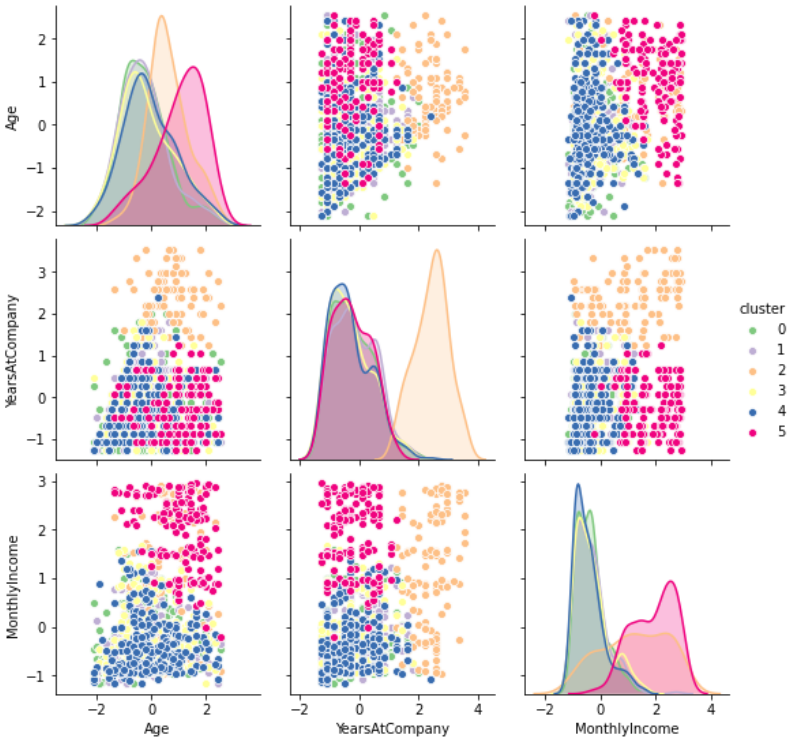


Figure 2.5 *Clustering for Age, YearsAtCompany and MonthlyIncome*

Figure 2.6 presents the centroid's analysis. The trend of cluster 0 is quite linear with similar (low) values for each attribute taken in consideration, and it has also a larger number of observations (Table 2.1). A different situation is associated with the other clusters. Regarding *YearsAtCompany* most of the clusters are focused on the same values, except from the cluster 2 that presents a higher value. Another interesting observation is the trend of clusters 0 and 4 which follow roughly the same pattern for all the attributes except for *YearsInCurrRole* where they separate themselves; this means that for a lower value of K they could merge.

Comparing figures, it is possible to notice that the centroids that are close to each other in figure 2.6 appear on top of each other in the scatter matrix visualization (figure 2.5). This is the effect obtained by the dataset analysed.

Table 2.1 *Count of observations for each cluster*

Cluster	Count of observations in each cluster
0	416
1	259
2	120
3	240
4	264
5	136

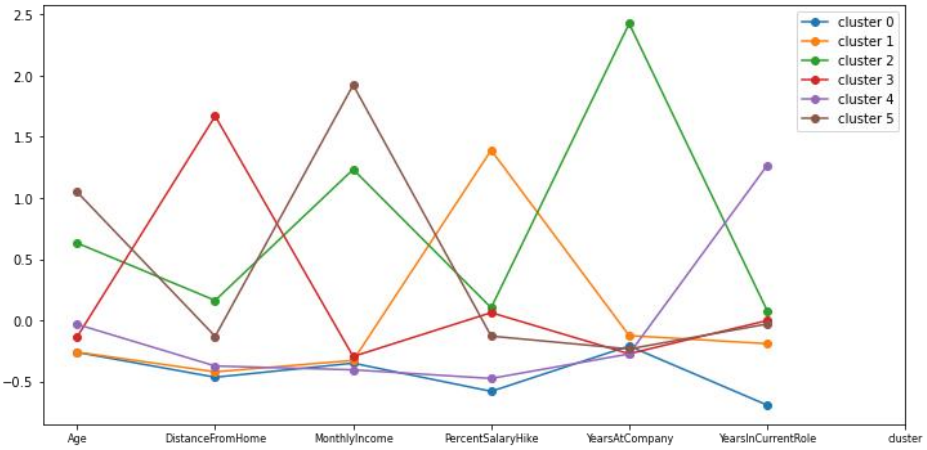


Figure 2.6 *Centroids analysis*

2.2 Analysis by density-based clustering

DBScan algorithm groups together the points of a dataset that are close to each other (neighbours), marking as noise points the ones that lie alone in low-density regions.

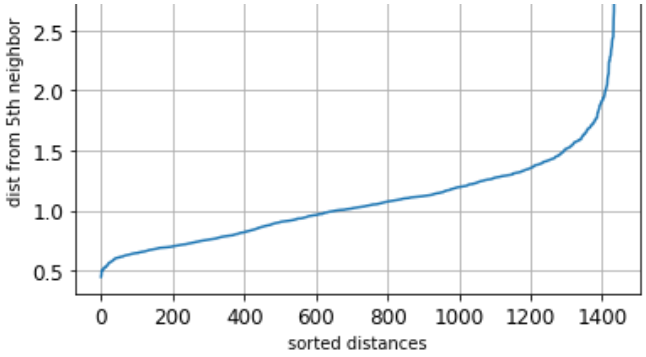
2.2.1 Distance function

As a distance function, the Euclidean distance, which is also the default metric of SkLearn, has been used to implement this algorithm. The process has been performed many times with different values of Epsilon and MinPoints.

2.2.2 Study of the clustering parameters

The main parameters to be considered when compiling a DBScan algorithm are ε (eps), which defines the radius of neighbourhood around a point x and the minimum number of points required to form a dense region (MinPts). For this dataset, as shown in the figure 2.7, the optimal value of eps has been calculated and found in the interval [1.4,1.7] (knee of the curve).

Figure 2.7 *Mean of distance between each point and the 5th nearest neighbors*



2.2.3 Characterization and interpretation of the obtained clusters

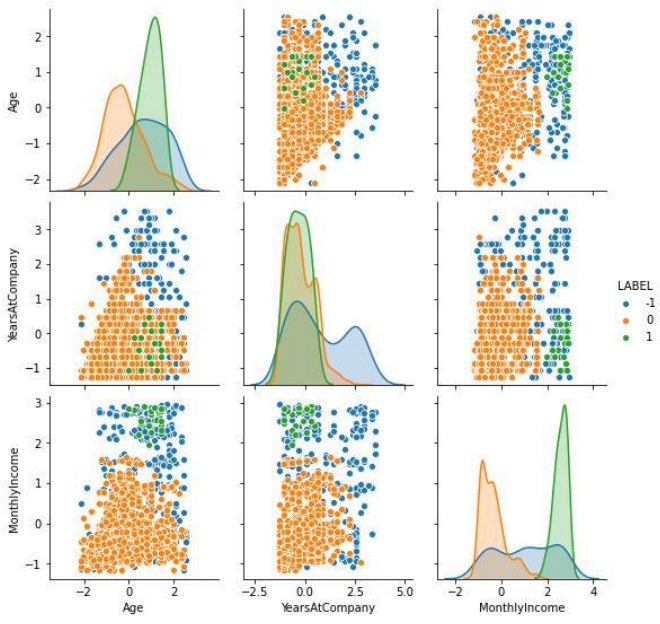


Figure 2.8 *DBSCAN clustering*

It has been decided to repeat the algorithm several times with different values of the parameters and study the different clusters obtained. As an example, when the MinPts value is equal to 50 the same results were found for each considered value of eps: a single cluster with noise points around it. As the value of MinPts decreases, the number of clusters returned by the algorithm increases, e.g. with MinPts = 20 and eps = 1.5 are present 2 clusters one bigger than the other and a decreasing number of noise points. The same pattern can be observed by keeping on decreasing the value of MinPts and varying the eps value.

After many iterations it has been decided to keep the clustering obtained with parameters **MinPoints = 20** and **Epsilon = 1.4**, a graphical visualization can be seen on the left side (figure 2.8).

2.3 Analysis by hierarchical clustering

Hierarchical Clustering method builds up clusters step by step. The size of the cluster is determined from the results at each step that are visually represented in a tree structure, dendrogram. The analysis has been implemented following an agglomerative process.

2.3.1 Distance function and dendrograms

The analysis has been done considering *Euclidean* distance and *Manhattan* distance. These two metrics have been applied to *complete*, *average* and *ward* clustering methods.
Since two different metrics have been used, a comparison has been implemented between the three methods, based on Silhouette performance. The table 2.2 shows that, according to the silhouette results, using *Euclidean* distance as metric Average seems to work better than the other methods.
In the following step, in table 2.3 the *Manhattan* metric has been considered. Also, in this case, the Average method seems to work in a more efficient way. For the Ward linkage a *none* value is reported because it works only with Euclidean metric.

Euclidean metric	Silhouette
Complete linkage	0.098
Average linkage	0.255
Ward linkage	0.156

Table 2.2 *Silhouette for Euclidean metric*

Manhattan metric	Silhouette
Complete linkage	0.112
Average linkage	0.224
Ward linkage	*None*

Table 2.3 *Silhouette score for Manhattan metric*

The number of clusters obtained are suggested by the default formula of colour threshold used by *scipy* python library.
For a better understanding, the figure 2.9 shows the dendrogram obtained with the Average method and *Euclidean* metric for distance. At a distance equal to 4.3 are obtained 3 clusters, and it is possible to observe that the next cluster is merged around 4.5 and the last one at the distance 6; probably the unique blue vertical line (on the left of the dendrogram) is a set of observations that could be considered outliers because of the greater distance.
On the other hand, when *Manhattan* metric is used, as a logical consequence of it, the distance between the clusters is higher (y axis). The lower silhouette score associated with *Manhattan* metric can be justified by the presence of 4 clusters.

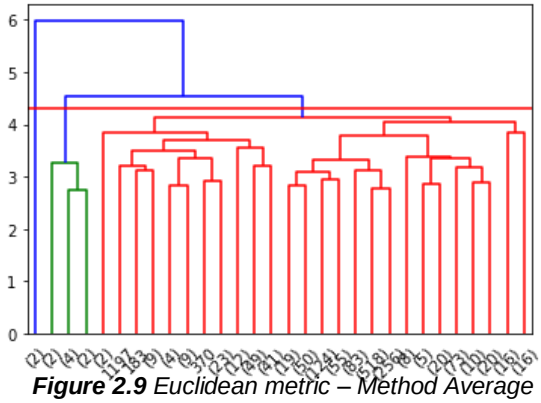


Figure 2.9 *Euclidean metric – Method Average*

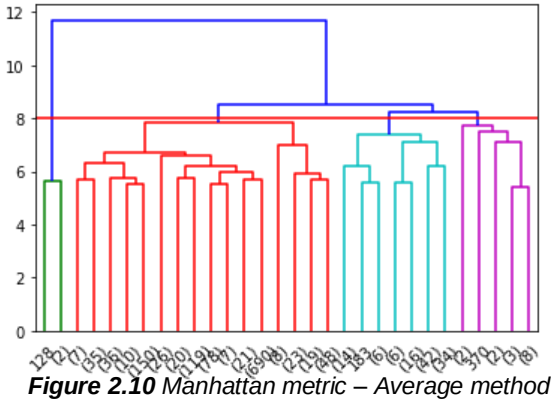


Figure 2.10 *Manhattan metric – Average method*

2.4 Evaluation of the best clustering approach

To evaluate the best clustering approach, it has been decided to consider the silhouette curve of the different clustering algorithms computed in the previous chapters. Table 2.4 compared the silhouette for K-means, DBScan and Hierarchical algorithms.

Clustering method	Silhouette score
K-Means	0.1853
DBScan	0.2192
Hierarchical	0.255

Table 2.4 *Silhouette for each clustering method*

To summarize, the values of the parameters used for each algorithm are:

- For the K-Means is used K = 6
- For DBScan are used Epsilon = 1.4 and MinPoints=20
- For the hierarchical method is considered the silhouette obtained by average method-Euclidean metric with 3 clusters

As a conclusion, given the characteristics of the dataset and the empirical results attained, it is possible to affirm that the method that fits better the data is the Hierarchical clustering.
However, also considering the visualization of the various clustering methods it does not seem to exist a clear choice for the best clustering for the dataset.

Association Rules Mining

In this section it will be described the association rules' analysis process.
Firstly, the dataset has been manipulated to prepare the data to the sequent operations: frequent pattern extraction (including maximal and closer frequent itemsets) using different values as threshold of minimum support [and minimum number of objects composing itemsets]; and after, extraction of the meaningful rules considering different level of confidence and lift. Finally, the most interesting rules are used to predict the target variable. Having treated missing values before, it has been decided to create random missing values to replace them using Association rules extracted. The results were not satisfactory, so this part has not been included in the report.

3.1 Data preparation

It has been needed to manage the data to make a correct association analysis. Going in order, firstly continuous attributes are been splitted into intervals (the size of the intervals will be different according to the domain of the attribute taken into consideration); the second transformation involves categorical data that during the analysis has been transformed into discrete (numeric) values, now such data have been restored to

their original categorical value; lastly, for a better understanding of the rules subsequently extracted, abbreviations have been added at the end of each value to recognize the attribute at which they refer.

The different ranges for continuous columns are chosen as below:

- *Age:* **[**<=30, >30 <=36, >36 <=43, >43**] _Years**
- *Monthly income:* **[**[5756.5, 10504.0), [1009.0, 5756.5), [10504.0, 15251.5), [15251.5, 20017.99)**] _USD**
- *DistanceFromHome:* **[**[8.0, 15.0), [1.0, 8.0), [15.0, 22.0), [22.0, 29.028)**] _Mile**
- *PercentSalaryHike:* **[**[14.5, 18.0), [11.0, 14.5), [21.5, 25.014), [18.0, 21.5)**] _%**

3.2 Frequent pattern

The Frequent Patterns extraction and the Association Rules extraction have been carried out by means of the *Apriori algorithm* implemented in the Pyfim library. After some testing, it has been decided to consider only itemsets with a minimum length of 4 elements, with the aim of getting the most significant ones.
Different values of support have been tested (10%, 20% and 30%) considering different types of itemsets (frequent, closed, and maximal).
Table 3.1 shows the number of extracted itemsets of length 4 by support and by type. Since changing the support

Support	Frequent I.	Closed I.	Maximal I.
10%	3287	3244	1536
20%	164	164	102
30%	11	11	11

Table 3.1 – *Count of itemsets extracted*

Considering the relationship between frequent, closed, and maximal itemsets, increasing the support the number of itemsets extracted converge to the same value (sup = 30%); Instead, lower values of support (10-20%) highlight the fact that closed I. and maximal I. are subsets of the frequents ones. In order to compare performance between the different types, the itemsets discovered have been sorted.

As it is possible to see from the table below (3.2), the traits that appear mostly frequently with employees that do not leave the job (attrition: No) are: Travel rarely, No Overtime and Performance rating at excellent; it's interesting to notice that comparing the first maximal itemset for support equal to 30% -keeping it fixed, 'Research and Development' and 'Male' are the features that are added due to the decreasing of support level. Since the dataset is very unbalanced towards employees who have not left their job, itemsets that involve attrition Yes (considering the support level used above) were not discovered.

Sup	Frequent itemsets	Maximal Itemsets
10%	1. ('Travel_Rarely', 'OT: No', 'No', 'PRat: Exc') 2.('Research & Develop', 'OT: No', 'No', 'PRat: Exc') 3.(‘'Research & Develop', 'Travel_Rarely', 'No', 'PRat: Exc) 4.('WLB: Better', 'OT: No', ' No', 'PRat: Exc’) 5.('[1009.0, 5756.5) _USD', 'OT: No', 'No', 'PRat: Exc')	1.(('Male', 'Research & Development', 'OT: No', 'Travel_Rarely', 'No', 'PRat: Exc') 2.(('JobLevel: 2', 'OT: No', 'Travel_Rarely', 'No', 'PRat: Exc') 3. (('Male', 'WLB: Better', ‘OT: No', 'Travel_Rarely', 'No', 'PRat: Exc') 4.(('[1.0, 8.0) _km', 'Research & Develop', 'OT: No', 'Travel_Rarely', 'No', 'PRat: Exc') 5.(('StockOpt:1', 'JobInv: High', 'OT: No', 'No', 'PRat:Exc')
20%	1. ('Travel_Rarely', 'OT: No', 'No', 'PRat: Exc') 2.('Research & Develop', 'OT: No', 'No', 'PRat: Exc') 3.(‘'Research & Develop', 'Travel_Rarely', 'No', 'PRat:Exc) 4.('WLB: Better', 'OT: No', ' No', 'PRat: Exc’) 5.('[1009.0, 5756.5) _USD', 'OT: No', 'No', 'PRat: Exc')	1.(('Research & Develop', 'OT: No', 'Travel_Rarely', 'No', 'PRat: Exc') 2.(('Married', 'OT: No', 'No', 'PRat: Exc') 3.(('[11.0, 14.5) _%', 'Research & Develop', 'No', 'PRat: Exc') 4.(('JobInv: High', 'Male', 'No', 'PRat: Exc') 5. (('JobInv: High', 'WLB: Better', 'No', 'PRat: Exc')
30%	1. ('Travel_Rarely', 'OT: No', 'No', 'PRat: Exc') 2.('Research & Develop', 'OT: No', 'No', 'PRat: Exc') 3.(‘'Research & Develop', 'Travel_Rarely', 'No', 'PRat:Exc) 4.('WLB: Better', 'OT: No', ' No', 'PRat: Exc’) 5.('[1009.0, 5756.5) _USD', 'OT: No', 'No', 'PRat:Exc')	[Same as frequent]

Table 3.2 *– Frequent Itemset and Maximal Itemsets sorted by support*

3.3 Association rules extraction

As far as the Association Rules extraction is concerned, different combinations of support and minimum confidence have been tested; support may assume the values 20% and 10%, while the minimum confidence values considered are 70%, 60% and 50%, moreover the minimum number of elements that compose the rules considered is three. Only the rules having lift greater than or equal to 1 are considered, since they are the most valuable ones.

For this reason, most of the rules extracted have been ignored, besides are shown the count for the rules discovered for all possible levels of support and confidence.

Regarding the last two columns of both tables, have been considered all the rules containing attrition Yes (or Attrition No respectively) as precedent or consequent. Going into detail of the rules with support 10% and confidence 50% that have attrition Yes, it is

Table 3.3 – *Rules extracted with support 10%*

Support = 10%				
Min. conf.	# Total Rules	# Rules (lift >= 1)	#Rules (Attr: No)	#Rules (Attr: Yes)
70%	14846	11179	6645	1
60%	31407	22935	12020	6
50%	50507	29631	14811	12

Table 3.4 – *Rules extracted with support 20%*

Support = 20%				
Min. conf.	# Total Rules	# Rules (lift >= 1)	#Rules (Attr: No)	#Rules (Attr: Yes)
70%	1712	1284	771	0
60%	3673	2670	1462	0
50%	5904	3506	1815	0

possible to note that one of them has Attr: Yes as its precedent, and the remaining as their consequent. They are shown in the table below.

#	Rules: **Yes-> (...)**	Confidence	Rules: **(…)->Yes**	Confidence
1	('Yes', ('OverTime:Yes', 'JobLevel: 1')	52.9 %	('PRat:Excellent', ('Yes', 'StockOpt:0')	87%
2			('PRat:Excellent', ('Yes', 'Travel_Rarely')	85.7%
3			('Single', ('Yes', 'StockOpt:0')	77.9%
4			('Travel_Rarely', ('Yes', 'PRat:Excellent')	72.7%

Table 3.5 – *Extracted rules having Attrition yes as precedent and consequent*

3.3.1 Target variable prediction

Let's consider the rule {'PRat: Excellent', 'StockOpt:0} -> 'Yes'; it is applicable to 520 records in the dataset, classifying them as leaving employees, but only 134 of them are actually employees who left, so there are 386 false positives; on the other hand, the rule is not able to classify as leaving employees, the other 106 (237-131) employees that actually left on which the rule is not applicable. The same procedure can be applied to one of the rules that have 'Attrition No' as consequent, like {'PRat:Excellent','Travel_Rarely'} -> 'No', it has been executed on 875 records, 731 of them have been correctly classified. The final evaluation of performance of these rules is summarized by table 3.6.

Table 3.6 *– Evaluation of predictive model*

TP: 134	**FP:** 386
FN: 144	**TN:** 731
Accuracy = 0.62	

The final consideration is about the unsuitability of extracted rules to predict the entire dataset since they have very low level of support (little significancy). So, it is possible to say that the extracted rules alone are not good enough to build a good overall rule-based classifier.
Moreover, it could be interesting to visualize the confidence and lift values for all the rules mentioned before graphically.

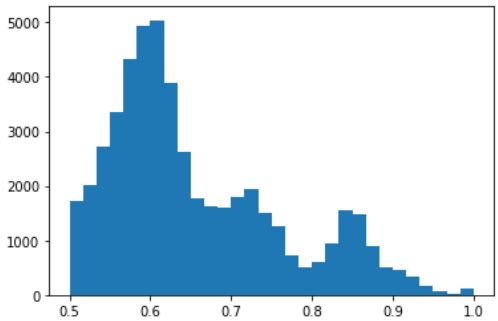


Figure 3.1 *– Histogram of rule's confidence level*

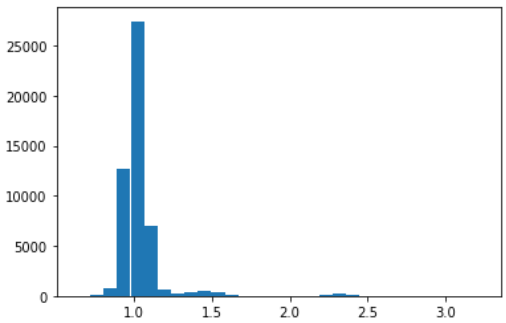


Figure 3.2 *– Histogram of rule's lift level*

Classification

The classification task has been carried out by means of the *Sklearn* library in Python 3.7. Due to the merge done in the Data Understanding section, the dataset was splitted into train and test (using a percentage of splitting equal to 20%). Furthermore, in order to have a validation set where to implement intermediary analysis regarding the efficiency of the classifiers, the train set was split again into train and validation (now using 25% as splitting percentage). The models tried were *Decision tree Classifier*, *Random Forest Classifier* and *KNN*.
Firstly, the models were built on the original (unbalanced) dataset tending to classify few employees who left the job (Attrition = Yes). Due to these outcomes resulting from the imbalance of the dataset on the target variable (Attrition), it was deemed necessary to apply Undersampling and Oversampling techniques to rebalance the data.
So, in the following sections the analysis was carried out comparing the results obtained with the training dataset unbalanced to the balanced ones, in order to determine the best model.

4.1 Data preparation

As first point the type of categorical attribute was changed mapping it into integer number. It permits to use them with the decisional tree algorithm.
A 50% *random over sampling* strategy has been applied on the training set by resizing the target class so that the minority class reaches the proportion established. Therefore, the training dataset will have 1148 records ('No': 959, 'Yes': 189).
To achieve the desired class distribution, *random under sampling* is used for selecting examples from the majority class and deleting them from the dataset. So, the final shape of the dataset is 958 records, where both classes have 479 observations.
It should be noted that the change to the class distribution is only applied to the training dataset, influencing the fit of the model. Furthermore, the choice to use these techniques impose an important bias because it could change the original nature of data distribution.

4.2 Decision Tree Learning and Validation

The first goal to reach in the analysis was the tuning of parameters that maximize the performances. To do that, a grid search algorithm was implemented with different values. For both, unbalanced and balanced data, the following range of parameters are tested, in detail: Depth a range between 1 and 20, min samples split in range {2, 5, 10, 20, 30, 50, 100} and min samples leaf in range {1, 5, 10, 20, 30, 50, 100}. That task was implemented using a value for k-fold equal to 4.
The table in the next page (table 4.1) shows the rank of the set of hyperparameters and their relative mean validation score. *Grid Search* gives the same rank for all sets, and the same validation score, it means that fixing max_depth and min_sample_leaf, any value assumed by min_sample_split does not change the result. In the light of this, Depth: 3, min_samp_split: 10, min_samp_leaf: 30 are chosen as parameters, which performed mean validation score: 0.855 (std: 0.010).

#	Parameters	Mean validation score
1	{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 30, 'min_samples_split': 2}	0.855 (std: 0.010)
1	{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 30, 'min_samples_split': 5}	0.855 (std: 0.010)
1	{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 30, 'min_samples_split': 10}	0.855 (std: 0.010)

Table 4.1 - *Hyperparameters found by grid search algorithm on unbalanced dataset*

Looking at the figure 4.1, a graphical representation of the most important features is given. *StockOptionLevel*, *JobLevel*, and *OverTime* play a dominant role, it means how the splitting conditions on that attributes are relevant for the aims of the analysis.
As it will show in the decisional tree representation, the difference condition of work and the attribution of bonus should be taken in consideration as important factors for employees who left the company.
Figure 4.2 shown the resulting tree given by the classifier.

Figure 4.1 *– Features importance*

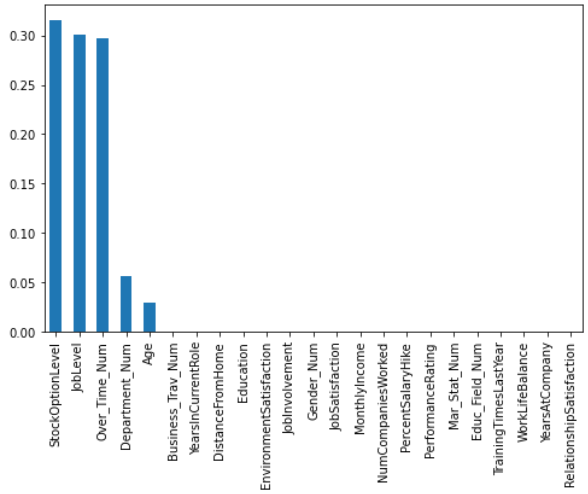
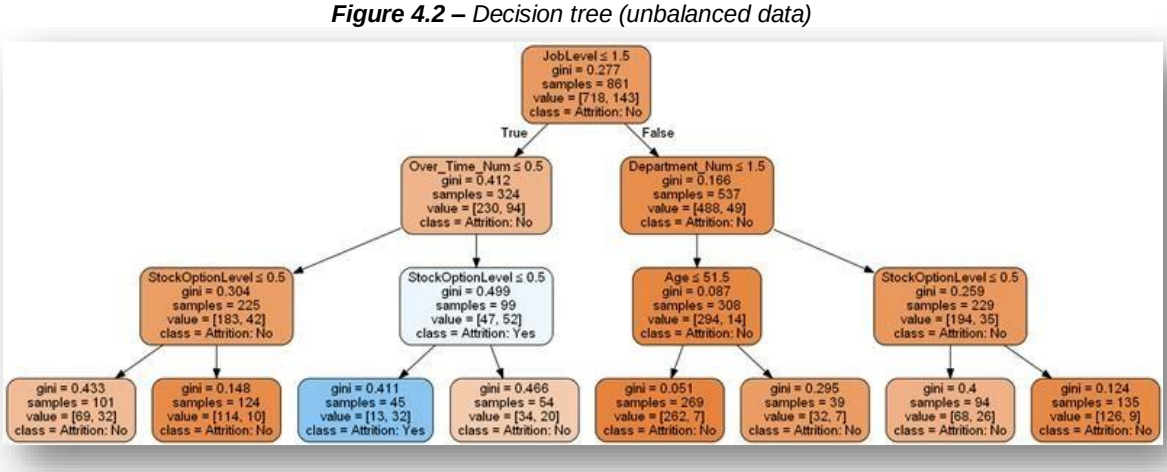


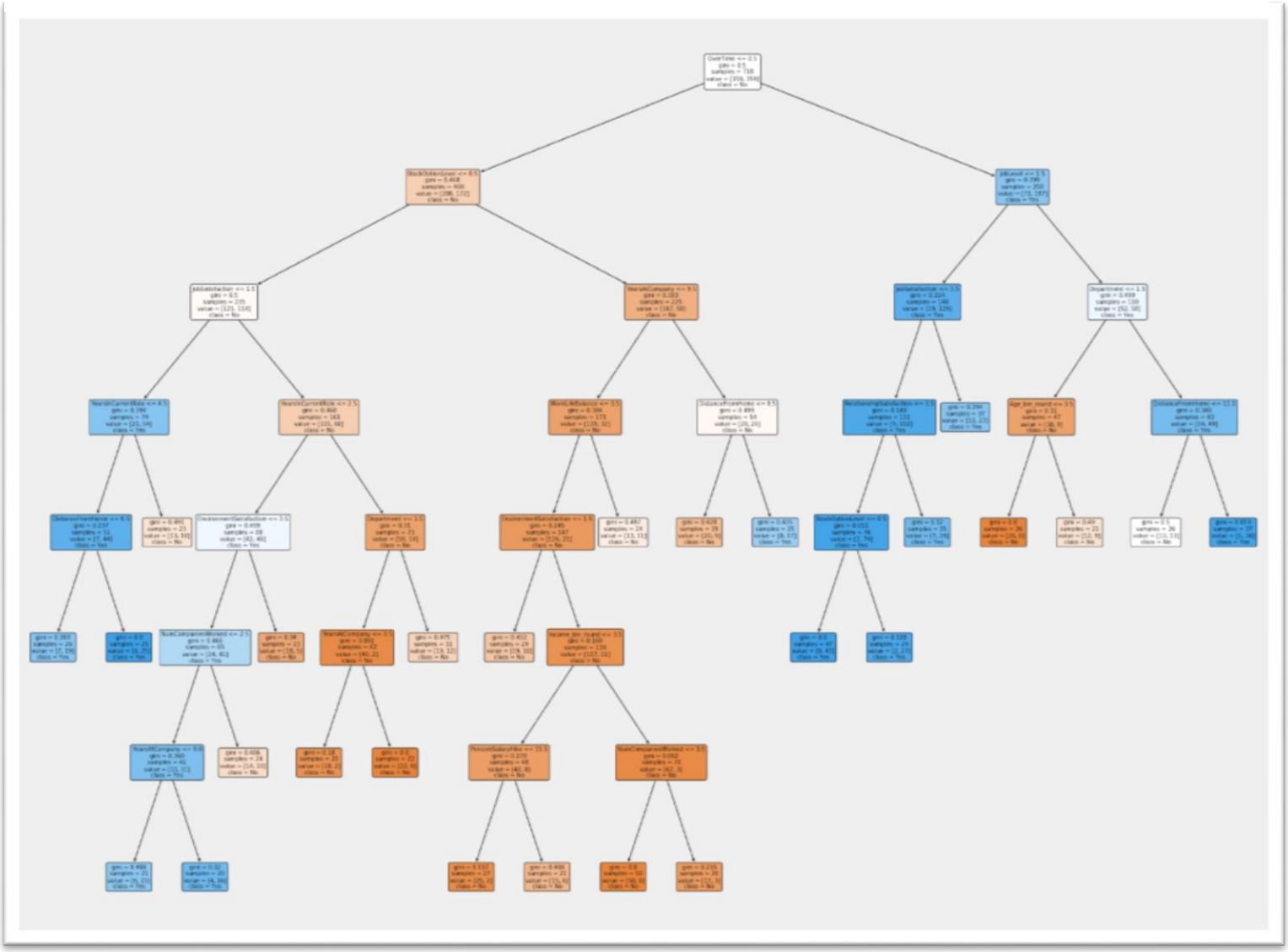
Figure 4.2 – *Decision tree (unbalanced data)*



4.2.1 Balanced Dataset

Using the same algorithms to choose the best hyper parameters, Decision tree Classifier has been implemented on the balanced training set. Using (criterion = 'gini', max_depth =8, min_samples_split = 2, min_samples_leaf = 20). The decision tree built is shown below (figure 4.3); it includes for each node the splitting condition associated with the related feature, the impurity measure chosen (gini index) and the number of observations included on splitting.

Figure 4.3 – *Decision Tree (Balanced dataset)*



4.3 Decision Tree Interpretation and evaluation

Comparing the two models before, the differences between them are of immediate understanding in terms of graphical visualization. The first decision tree is less deep, and it includes only few attributes for splitting conditions, on the other hand the balanced decision tree seems to be more accurate on classification considering more attributes as splitting conditions (it is due the greater depth). Anyway, the final -graphical- evaluation could be pretty the same: the conditions that contribute most to leaving the job are if an employee is involved in overtime, if him/her has a low job level and their stock option level.

More in detail, to establish which is the model that performs better, it is necessary to compare the evaluating measures as Accuracy, F1 Score, Precision and Recall of both.

By the comparison of the tables 4.2 and 4.3, as first impression, the model built in the unbalanced training set feels to perform better. It shows in the validation set an accuracy value (0.85) greater than the classifier built in the balanced training set (0.77). Going more in depth, and thinking to the purpose of the study, the unbalanced model is quite perfect in predicting No Attrition values but has poor score in classifying employees who left the company, the recall value is low to be acceptable. In contrast, the balanced model has

a lower precision and recall concerning No Attrition values but shows a significant improvement in correctly classifying employees who leave the company (recall 0.69).

BALANCED training set					**Validation Set**				
	Precision	Recall	F1-Score	Support		Precision	Recall	F1-Score	Support
No	0.75	0.87	0.81	359	**No**	0.73	0.84	0.78	120
Yes	0.85	0.72	0.78	359	**Yes**	0.81	0.69	0.75	120
Accuracy	**0.80**				**Accuracy**	**0.77**			

Table 4.2 – *Performance metrics of balanced decision tree classifier*

UNBALANCED training set					**Validation Set**				
	Precision	Recall	F1-Score	Support		Precision	Recall	F1-Score	Support
No	0.86	0.98	0.92	718	**No**	0.86	0.98	0.92	240
Yes	0.71	0.22	0.34	143	**Yes**	0.64	0.19	0.30	47
Accuracy	**0.80**				**Accuracy**	**0.77**			

Table 4.3 *– Performance metrics of unbalanced decision tree classifier*

Another evidence, concerning the preferability of a balanced model is given by the Roc curves (figures 4.4 and 4.5). Analysing it in terms of sensibility and specificity, balanced model presents a higher level of sensibility at a higher level of specificity. Due to this evidence, it was decided to continue the analysis and the comparison with the other models using the balanced classifier.
A double analysis was done between balanced and unbalanced training sets for all the following models. To be more synthetic only the balanced case is reported since is the best.

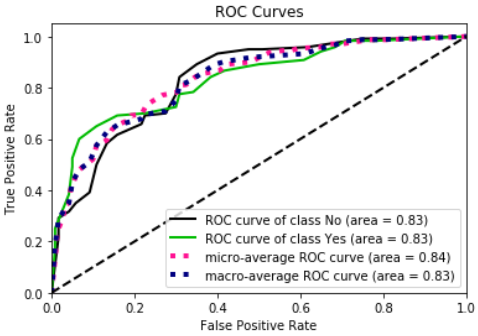


Figure 4.4 – *ROC Curves of balanced classifier*

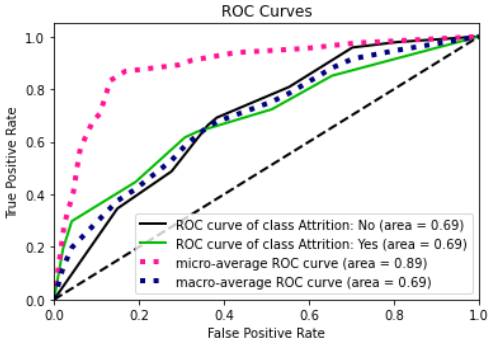


Figure 4.5 – *ROC Curves of unbalanced classifier*

4.4 Random Forest classifier

The Random Forest is an ensemble algorithm which was used in this case to combine different decisional tree models. As the first point, *randomized search* was used to determine the best parameters used into the implementation. The parameters chosen were *Criterion*: Entropy, *Max_Depth:* None, *Min_samples_leaf*: 5, *min_samples_split*: 10 with a mean validation score of 0.85 (std: 0.021). The evaluation was made computing the accuracy in training and validation sets as before. The RF classifier applied on training predicted well 348 records as Attrition No (True Negative) and misclassified 11 records as False Negative; on the other hand, it

well classified 343 records as Attrition Yes (True Positive) and it misclassified 16 records as False Positive. On the validation set it predicted TN:100, FN: 20, TP: 101, FP: 19. The **train** accuracy is **0.94 ,** its **f1 score** is **0.94**; the **validation** Accuracy is **0.81** and its **f1 score in range [0.79, 0.82].** The measures found in the training set could suggest a situation of overfitting. Below it is presented the figure (4.6) of the values of ROC Curve and the tables concern the precision, recall measures; they seem to confirm the precedent consideration.

Training set					**Validation Set**				
	Precision	Recall	F1-Score	Support		Precision	Recall	F1-Score	Support
No	0.94	0.96	0.95	359	**No**	0.86	0.74	0.79	120
Yes	0.96	0.94	0.95	359	**Yes**	0.77	0.88	0.82	120
Accuracy	**0.94**				**Accuracy**	**0.81**			

***Table 4.4** – Performance metrics of Random Forest classifier on training and validation sets*

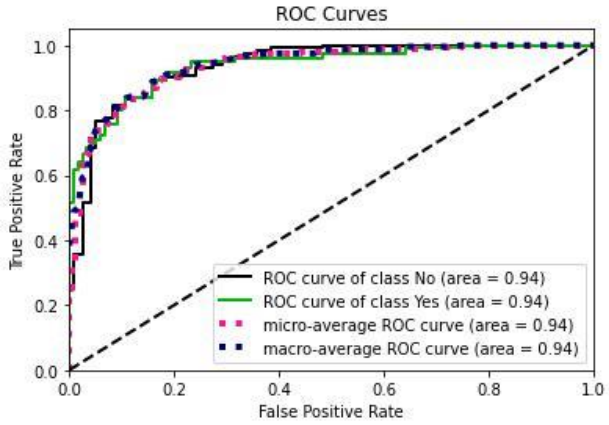


***Figure 4.6 –** ROC Curve of Random Forest Classifier*

4.5 KNN

The KNN classifier is an algorithm which permits to classify test instances based on the majority class of its nearest neighbours. To find the best number of neighbours, several parameters were tried also comparing the difference between weighted and unweighted implementation. The results show that setting weight equal to *distance* for all number of neighbours tried, the accuracy and the F1 Score are equal to 1; there could be risk that the model is in overfitting. Three implementations were done with the following output:

- **Neighbours = 5, weights = uniform -> accuracy = 0.7924, f1score= [0.77, 0.80]**
  weights=distance -> accuracy = 1, f1score= 1
- Neighbours = 10, weights = uniform -> accuracy = 0.7465, f1score = [0.73, 0.75]
  weights= distance -> accuracy= 1, f1score = 1
- Neighbours = 15, weights = uniform -> accuracy = 0.7061 , f1score = 0.70
  weights= distance -> accuracy=1, f1 score = 1

According to the results, the parameters chosen were highlighted, and the relative performance are summarized by the table 4.4 below.

Training set					Validation Set				
	Precision	Recall	F1-Score	Support		Precision	Recall	F1-Score	Support
No	0.83	0.73	0.78	359	**No**	0.64	0.60	0.62	120
Yes	0.76	0.86	0.80	359	**Yes**	0.62	0.66	0.64	120
Accuracy	**0.79**				**Accuracy**	**0.6375**			

Table 4.4 - *Performance metrics of KNN classifier*

4.6 Comparison between the proposed models

A comparison was done for all the proposed models. In order to choose the best classifier and confirm its efficiency in classification of the test set, evaluation with respect to validation sets was taken into consideration. In the context of the analysis' purpose, the accuracy and the recall are the performance metrics used to evaluate the predictors, in such a way so that to have a model that correctly predicts when an employee leaves the job (attrition yes). The model that best performed according to these parameters is the **Random Forest Classifier**, as previously seen the evaluation metrics are equal to **Accuracy 0.81** and **recall [Yes]** equal to **0.88**.

On the test set the following metrics were scored by the model, the results are shown in table 4.5 and confusion matrix 4.6.

This model, applied to a set of data never seen before, performs well enough if we consider the correctly classified record with the target variable Yes. In fact, it correctly predicted 30 record on 48 as Yes and 169 records on 240 total records as No. Instead, if we consider the precision it is less accurate, but we do not focus too much on that because it is preferred to classify more "yes" (False Yes + True Yes) than to classify "true yes" as "false no".

Test set				
	Precision	Recall	F1-Score	Support
No	0.92	0.82	0.86	240
Yes	0.41	**0.62**	0.49	48
Accuracy	**0.7847**			

Table 4.5 *- Performance metrics of RF classifier on Test Set*

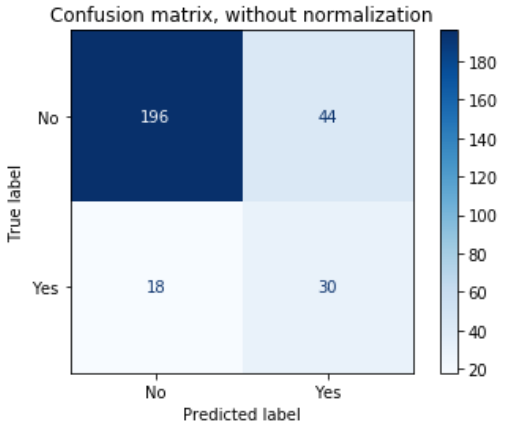


Figure 4.7 *Confusion matrix RF classifier on test set*