ASSESSMENT AND ANALYSIS OF LEVELS OF EMPLOYEE ATTRITION

STUDENTS GROUP

Cosimo Costantini, Samuele Cucchi,
Francesco Gemignani, Andrea Ribellino

DATA MINING PROJECT

Data Science & Business Informatics

Academic Year 2020/2021

Contents

1 Data Understanding

1.1 Data Semantics

The Dataset contains a collection of working and personal information regarding 1470 different employees working for IBM. The dataset is divided into a *training set* and a *test set*. Both of them are composed of 33 fields, respectively containing 1176 and 294 records, with a ratio equal to 80:20. Each field of the dataset can be divided into five different types of attributes:

- **Personal Information** - about the employees. Those values are: Age, Gender, MaritalStatus, Education and EducationField.
- **Attrition-Key status** - those fields describe the involvement and the emotive impact that an employee has, both related to his/her behavior on the working place and the relationship with other colleagues. Each variable can be *Low, Medium, High, Excellent*. Usually, low values could indicate an high level of attrition, but it's not always true. Those values are: EnvironmnentSatisfaction, JobInvolvement, JobSatisfaction, PerformanceRating, RelationshipSatisfaction and WorkLifeBalance.
- **Salary Amount** - numerical attribute representing the salary and eventual increments to it. Those values are: MonthlyIncome, HourlyRate, DailyRate, MonthlyRate and PercentSalaryHike.
- **Work History** - historical details relative to an employee that indicate the amount of years in which the employee has worked in the same company. Those values can be: YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCurrManager, TrainingTimesLastYear, NumCompaniesWorked and TotalWorkingHours.
- **Work Description** - categorical and numerical attributes describing the job of an employee, taking in account information like business travels, role, etc. Those values can be: DistanceFromHome, OverTime, JobLevel, StockOptionLevel, BusinessTravel, Department, JobRole, Over18 and StandardHours.
- **Attrition** - boolean value and variable to be predicted. It indicates whether the employee has been discharged from his job during the period when he/she was analyzed.

1.2 Distribution of the variables and statistics

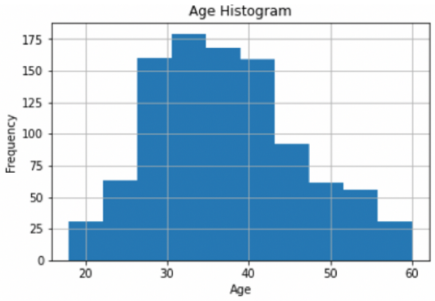
1.2.1 Numeric Attributes

Before evaluating or modifying the data quality, we analyzed the statistics of each group, in particular the mean (σ) and standard deviation (δ).

Age: discrete value that goes from 18 to 60. The histograms in figure 1.1 show a single distribution, with no outliers and an average of 37 years per employee.

Salary Amounts those are positive numerical values. XXXRate and PercentSalaryHike values are a well balanced distribution, furthermore, PercentSalaryHike indicates that 50% of the employees get a salary hyke equal to 11-14% of their incomes. Those attributes does not contain null values and, examining the boxplots is clear that there are no outliers. MonthlyIncome indicates an average value of 6565.9460, which can vary sensibly after the correction of the 213 missing values. Analyzing the distribution in figure 1.2 it is evident that the 50% of the employees has a maximum income equal to 4969, the rest of the values is distributed between 4969 to 19999, that, statistically speaking falls inside range contained between the median value and over the 75percentile + 1.5*IQR (upper extreme).

	Age
count	1176
mean	37.199000
std	9.015802
min	18.000000
25%	30.000000
50%	36.000000
75%	43.000000
max	60.000000



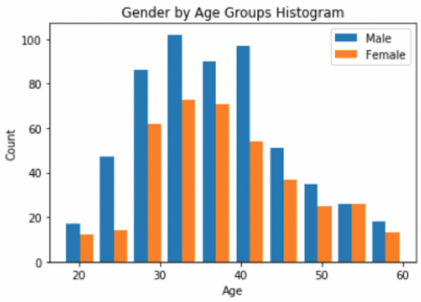


Figure 1.1: Age stats and histograms. On the right we also provides the number of attricted employees.

	HourlyRate	DailyRate	MonthlyRate	MonthlyIncome	PercentSalaryHike
count	1176	1176	1176	963	1176
mean	66.2993	803.6505	14395.8367	6565.9460	15.1768
std	20.2661	406.6830	7111.8451	4710.6256	3.6239
min	30.0000	102.0000	2097.0000	1009.0000	11.0000
25%	49.0000	460.5000	8227.2500	2969.0000	12.0000
50%	66.0000	804.0000	14434.0000	4969.0000	14.0000
75%	84.0000	1169.0000	20489.2500	8585.0000	18.0000
max	100.0000	1499.0000	26999.0000	19999.0000	25.0000

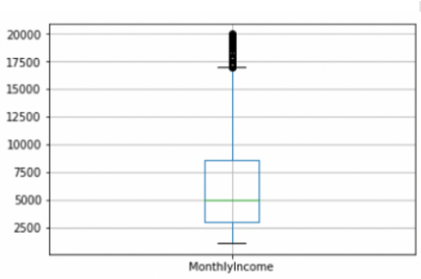


Figure 1.2: Salary amount group stats and MonthlyIncome boxplot in which appears outliers.

Work History variables are discrete numeric attributes which domain goes from 0 to 40 for YearsAtCompany and TotalWorkingYears. We see an unbalanced distribution: the standard deviation is close to the mean, due to the fact that the range of the possible values is little. Probably this is due to a limited set of outliers that could affect the measurements. The attributes TotalWorkingYears and DistanceFromHome show instead a more stable distribution, even if they contain some values after the median value.

	YearsAt Company	YearsIn CurrentRole	YearsSince LastPromotion	YearsWithCurr Manager	TotalWorking Years	NumCompanies Worked	TrainingTimes LastYear	Distance FromHome
count	1116	1176	1176	1176	1176	1176	943	1176
mean	6.9265	4.1887	2.1717	4.1079	11.0195	2.6632	2.8271	9.2100
std	6.0631	3.6374	3.1897	3.6010	7.6948	2.4912	1.2731	8.0970
min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000
25%	3.0000	2.0000	0.0000	2.0000	6.0000	1.0000	2.0000	2.0000
50%	5.0000	3.0000	1.0000	3.0000	10.0000	2.0000	3.0000	7.0000
75%	9.0000	7.0000	3.0000	7.0000	15.0000	4.0000	3.0000	14.0000
max	40.0000	18.0000	15.0000	17.0000	40.0000	9.0000	6.0000	29.0000

Figure 1.3: Work History group stats.

1.2.2 Categorical Attributes

Gender: binary attributes with 59 missing values (5.00%). In total there are 59.5% of men (Male) and 40.5% of women (Female) among all non-null values.

MaritalStatus: in this field, 46.1% are married (Married), 32.6% are Single and the remaining 21.3% are divorced (Divorced).

Education: attribute indicating the degree of education of each employee, into the figure 1.4 the distribution and the translation is shown.

EducationField: categorical attribute with 6 classes, consisting of 41.58% LifeSciences, 31.46% Medical, 10.62% Marketing, 9.09% Technical Degree, 1.78% Human Resources and 5.44% Other.

val	class	Distribution (%)
1	Below college	11.8197
2	College	19.8980
3	Bachelor	39.0306
4	Master	26.5306
5	Doctor	2.7211

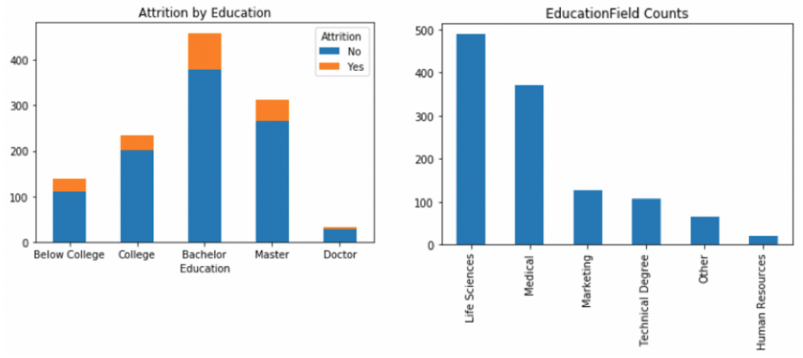


Figure 1.4: Education stats and bar charts.

Attrition-Key group is composed of 6 attributes with discrete values, each of which associates an integer value of the domain [1,4] to a rating class, respectively: Low, Medium, High and Excellent. There are no wrong values and no missing values inside this field.

	Enviroment Satisfaction(%)	Job Involvement (%)	Job Satisfaction (%)	Performance Rating (%)	Relationship Satisfaction (%)	Worklife Balance (%)
Low	19.2177	5.5272	19.9830	00.0000	19.3878	5.6973
Medium	19.8129	25.7653	19.9830	00.0000	20.7483	23.1293
High	29.8469	58.3333	29.8469	84.7784	30.1020	61.0544
Excellent	31.1224	10.3741	30.1871	15.2216	29.7619	10.1190

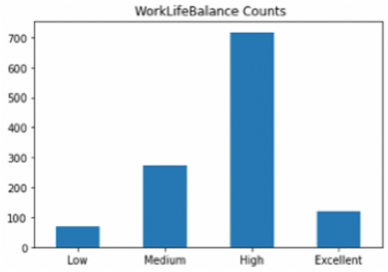


Figure 1.5: Attrition Key statistics and bar chart of WorkBalance.

OverTime: shows 71.25% of "No" and 28.75% "Yes".

BusinessTravel: contains 107 (9.09%) missing values. This record is composed by 71.47% "Travel Rarely", 17.96% "Travel Frequently" and 10.57 "Non-Travel" (considering only not-null values).

Department: attribute containing 3 types of sectors: 65.39% "Research & Development", 30.70% "Sales" and 3.91% "Human Resources".

JobRole and **JobLevel**: categorical and discrete ordinal respectively.

JobRole	Distribution (%)
Sales Executive	22.1939
Research Scientist	20.4082
Laboratory Technician	18.0272
Manufacturing Director	9.7789
Healtcare Rapresentative	9.0136
Manager	6.8878
Sales Rapresentative	5.8673
Research Director	4.3367
Human Resource	3.4864

JobLevel	Distribution (%)
1	37.925
2	36.479
3	14.881
4	6.972
5	3.741

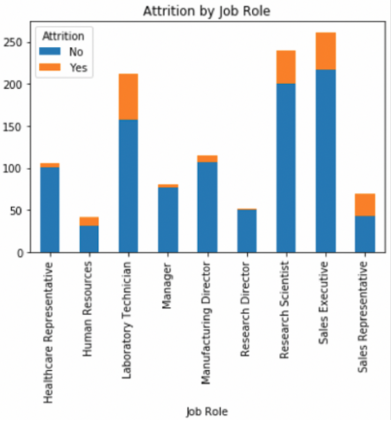


Figure 1.6: JobRole and JobLevel tables.

Over18: contains 372 missing values and has a single category with 804 'Y' values. It provides redundant

information, since we have the Age attribute.

StandardHours: Numeric attribute with 570 (48.47%) null values. All the others value are all set to 80. For the reasons explained before we decided to remove this field and Over18 in way to reduce the size of the dataset, bringing it to 31 attributes.

Attrition: categorical attribute and target variable. It is made up of 984 (83.67%) 'No' and 192 (16.33%) 'Yes'. The following cross-charts show the attrition value related with some other attributes. After normalisation, we can see the resignation index, stating that 47% of employees resign in the first 2 years of work and that a single worker has more probability to be dismissed respect a divorced or married employee. Alike a person who works for extra time.

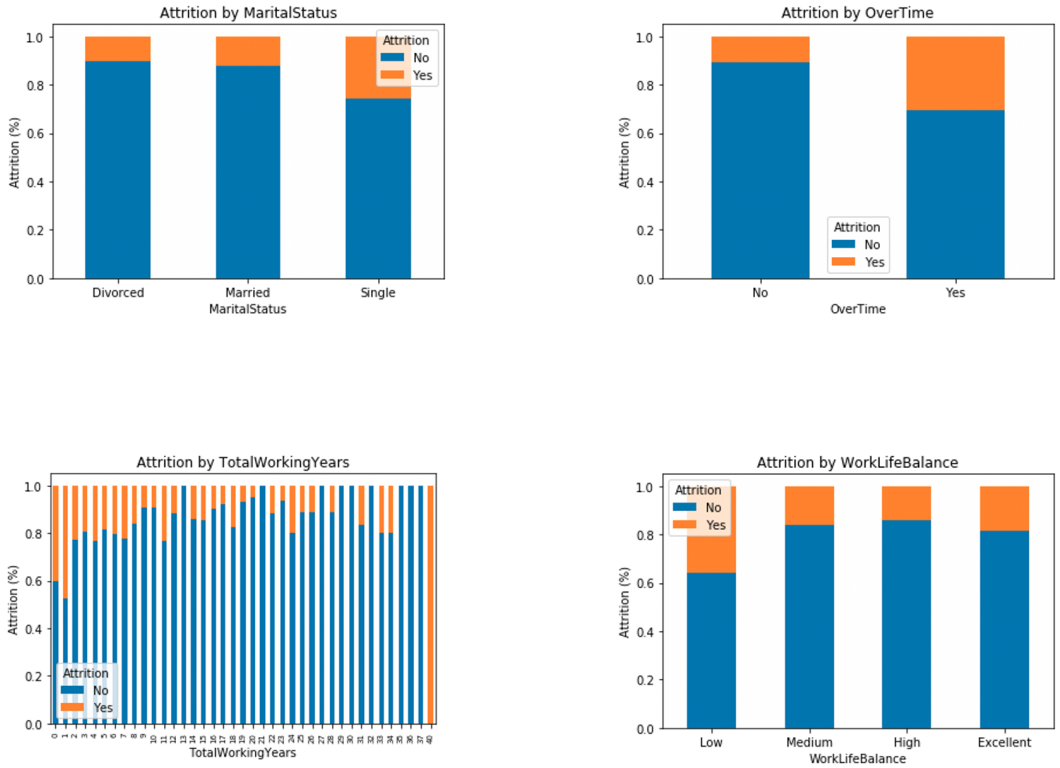


Figure 1.7: Attrition cross plots. In those plots the distributions are normalized. Therefore we can see the attrition frequence that an employee (among the total detected) has diversifying them basing on different values (values name listed on x-axis).

1.3 Evaluation of data quality and transformation variable

In this section we will describe how we managed missing values, outliers and possible dataset dimensionality reduction. Given the small amount of data that constitutes the dataset (1176 different employees) we have tried our best not to delete further records, trying to replace values with approximations.

1.3.1 Missing Values

Attribute	Age	Business Travel	Gender	Monthly Income	Over18	Performance Rating	Standard Hours	TrainingTime LastYear	YearsAt Company
Missing Values	176	107	59	213	372	138	570	233	60

Figure 1.8: Number of missing values

To decide how to treat attributes BusinessTravel, Gender and PerformanceRating, we plotted different types of graphs crossing different attributes between each other. To do this, we choose the mode of those

values because, independently from the attribute which has been compared with, it always remains the same for every subset. As a result, null values inside Gender attribute have been substituted with 'Male'. The ones in PerformanceRating with the value 3 (High) and inside the BusinessTravel attribute the Travel_Rarely value have been used.
Differently, for numerical attributes we focused on correlations. As we can see from the correlation matrix 1.12, the MonthlyIncome attribute has a correlation index of 0.50 related to YearsAtCompany. According to that, we decided to discretise the values of YearsAtCompany in four different intervals: each bin contains the values of a quartile. Then we computed the average salary by grouping MonthlyIncome according to the so defined intervals. At the end of the computation, the correlation index remained 0.50, and we obtained a decrease in the standard deviation by 8.29%.
We worked in the same way for the Age, grouping it on the quartile basis in relation with MonthlyIncome, whose correlation index is equal to 0.44. Also in this case we have maintained the same correlation index, and a significant lowering of the standard deviation. To sum up, we have chosen the quartiles to have a good 25% of the distribution on each quartile (294 records) to evaluate a reliable average value. A greater number of intervals would have further reduced the number of records to be computed, lowering the accuracy of the result. Furthermore, if we had considered intervals of equal width we'd have had empty bins or with a very limited distribution on which to calculate the average.

The TrainingTimeLastYears attribute is composed of 19.81% of null values and the domain is discrete with integers ranging from 0 to 6. Therefore, instead of considering the mode (with 48.63%) we decided to replace the null values with values ranging within the respective probabilities: i.e. 0 to 2.72%, 1 to 3.74%, 2 to 48.63%, 3 to 27.72%, 4 to 6.97%, 5 to 6.97% and 6 to 3.23%, to keep the distribution as stable as possible.

1.3.2 Outliers

Since the histograms of the categorical attributes did not show any isolated group, we plotted box-plots of the numerical attributes. This let us find the outliers shown in the following table, that have been counted as values of the distribution outside the intervals $Q1 - 1.5 * IQR$ and $Q3 + 1.5 * IQR$, using the IQR scores approach.

Attribute	Monthly Income	NumCompanies Worked	TotalWorking Year	TrainingTimes LastYear	YearsAt Company	YearsIn CurrentRole	YearsSince LastPromotion	YearsWith CurrManager
Outliers	89	40	49	152	75	17	85	11
Thresholds Q1-1.5*IQR, Q3+1.5*IQR	[-3343.875, 14775.125]	[-3.5, 8.5]	[-7.5, 28.5]	[0.5,4.5]	[-6.0, 18.0]	[-5.5, 14.5]	[-4.5, 7.5]	[-5.5, 14.5]
Domain	[0, 19999]	[0, 9]	[0, 40]	[0, 6]	[0, 40]	[0, 18]	[0, 15]	[0, 17]

Figure 1.9: Outliers count with IQR score approach. We have reported also the threshold and domain range. So we can see how many outliers are distributed in the domain range.

As we saw before, all the attributes except MonthlyIncome have a limited domain, ranging from 0 to 40 (i.e. TotalWorkingYears). MonthlyIncome's outliers are all above the $Q3 + 1.5 * IQR$ threshold and identify employees with an income between 14775.125 and 19.999. We evaluated the semantic accuracy of these values by considering the correlations with Age and YearsAtCompany equal to 0.50 and 0.51 respectively, coming to the conclusion that the anomalies do not represent errors but they simply are a group of employees, equal to 7.56%, with higher income. From the scatter plot, we can see that higher salary values correspond with employees with higher age. We decided though, to replace anomalies by assigning them an appropriate value. In the first analysis we replaced the outliers by assigning them the threshold value $Q3 + 1.5 * IQR$, but at the same time this extreme binning altered the distribution, as we can see in figure 1.10.

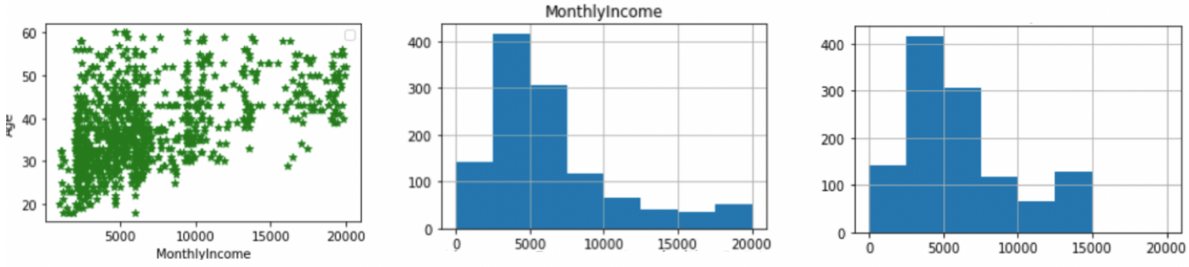


Figure 1.10: On the left the MonthlyIncome scatter shows a continuous distribution that leads to think that the outliers are not noise. The 2 histograms display as before and after replacing outliers with $Q3+1.5*IQR$ could alterate wrongly the distribution.

Therefore, we decided to apply a logarithmic transformation which, in addition to significantly improving the mean and standard deviation, brought the outliers back within the thresholds, without affecting the correlation at all.

	MonthlyIncomeTransf
count	1176
mean	8.59
std	0.61
min	6.91
25%	8.14
50%	8.56
75%	8.98
max	9.90

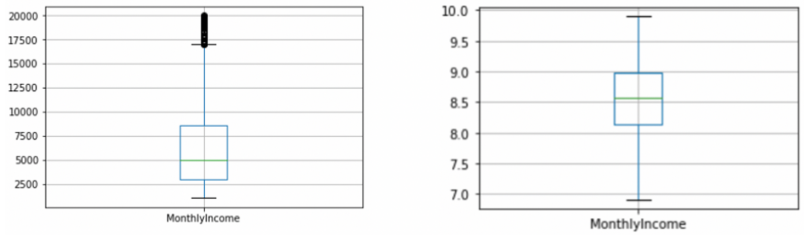


Figure 1.11: MonthlyIncome before and after logaritmic transformation.

The remaining attributes have a very narrow domain and a limited number of anomalies. We tried to eliminate outliers to understand how much the statistics would be improved, unfortunately the results weren't really satisfactory: the mean and the standard deviation didn't changed much. Considering the lack of correlation with other attributes and the limited number of records in the dataset, we decided not to eliminate those anomalies.

1.3.3 Pairwise correlations and dimension reduction.

The following table shows the correlation between attributes in the dataset after the quality assessment. The Over18 and StandardHours attributes have been removed because of their uselessness as they only contained a single domain value. Furthermore, Over18 is redundant because of the Age variable. In general, most of the attributes are very unrelated, except for some isolated cases such as JobLevel and TotalWorkingYears, whose size could also be reduced. The attributes YearsInCurrentRole, YearsSinceLastPromotion and YearWithCurrManager are redundant and have been merged on an average basis, inside the YearsMean class.

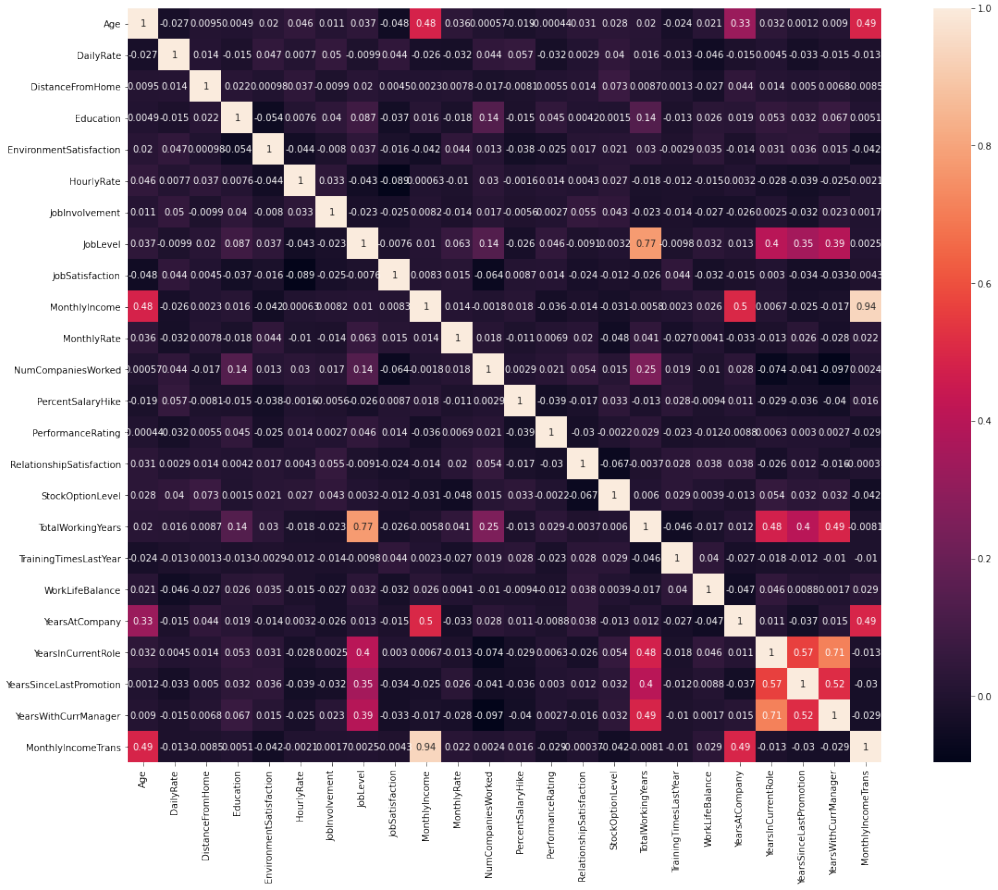


Figure 1.12: Correlation matrix. The gradient indicate more correlated attributes.

We decided, furthermore to insert the YearsMean field, which is an attribute containing the mean between the columns YearsInCurrentRole, YearsSinceLastPromotion and YearsWithCurrManager because they have an high correlation rate, as clearly shown in the correlation matrix. We decided not to remove the attributes discussed before because during clustering we will work with a subset of attributes.

2 Clustering

2.1 K-means

K-means is an iterative clustering algorithm that divides the dataset into K (pre defined) number of clusters, where each element of data is contained in only one cluster. This algorithm is based on the concept of keeping data points belonging to the same dataset as similar as possible, and data points contained into two distinct clusters, as different (far) as possible. K-means decides whether to assign a data point into a cluster or into another by calculating the sum of the squared distance between the data point under examination and each of the Ks clusters centroids.

Attributes and distance function

The attributes that have been used to compute the K-means algorithm are: Age, DistanceFromHome, MonthlyIncome, NumCompaniesWorked, TotalWorkingYears and YearsAtCompany. Those kind of value have been chosen because of their heterogeneity. As distance function we decided to use Euclidean distance.

Computing best number of k

In order to find the best number of Ks (centroids) in order to clusterize the dataset given we both used the Elbow method and the Silhouette score method.

Elbow method

The Elbow method gives an idea of which could it be the ideal number of clusters, this is done based on the Sum of Squared Errors (SSE) between data points and their clusters centroid. We'll pick the ideal value of K at the spot where the SSE starts to flatten out.

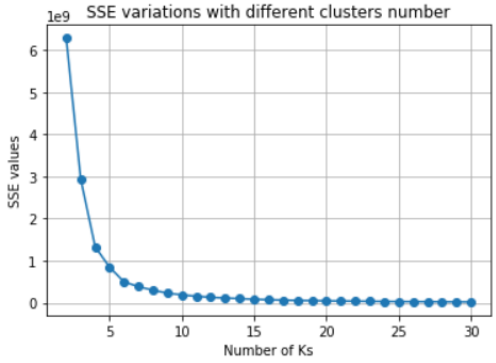


Figure 2.1: SSE plot for $k \in [2, 30]$

From the figure above we can state that a good number of clusters for this dataset can be 6, both because the SSE value is already pretty low and because it will not be a really demanding process to divide the dataset in six clusters.

Silhouette analysis

The Silhouette analysis is used to determine the degree of separation between clusters. This analysis is done by:

- Compute the average distance between points belonging to the same cluster.
- Compute the average distance between each point, and the points its cluster.
- Compute the coefficient by subtracting the first value computer by the second, all divided my the maximum value between the two.

The coefficient obtained will be in the interval [-1, 1];

- If *0* - the sample is very close to the neighboring clusters.
- If *1* - the sample is very far from the neighboring clusters.
- If *-1* - the sample is assigned to the wrong cluster.

Computing the Silhouette score taking into accounts value of k from 2 to 10, we obtained the data into Figure 2.2.

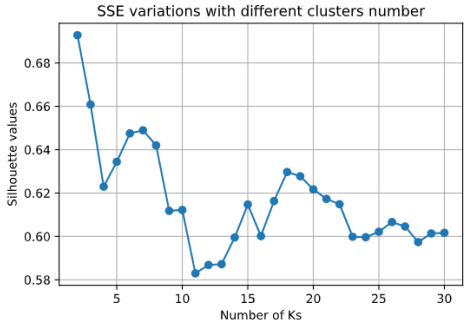


Figure 2.2: Silhouette plot for $k \in [2, 10]$

Examining the obtained graphs we can see that the best value of K is 2 because the silhouette average is the highest with such amount of Ks, but, since that we want to consider even the result obtained with KMeans, we decided to use 6 clusters.

2.1.1 Characterization and distribution of the clusters

The pairplot in the figure 3 represents the collection of all the scatter plot taking into account all the possible combinations of attributes.

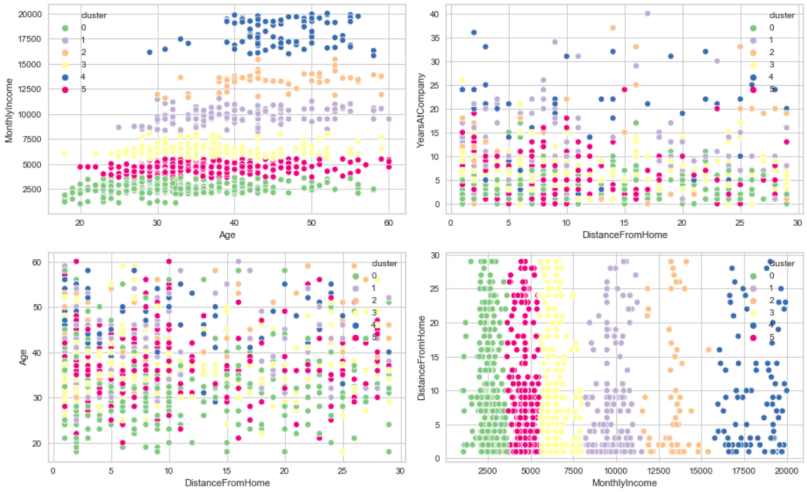


Figure 2.3: Scatter plot for different attributes.

As we can see from the scatter plots, a really dominant role into the division of clustering is made from the attribute MonthlyIncome.

Clusters description

- **Cluster 0** - Contains 315 elements, it is characterized from values having a mean age of 40.5, DistanceFromHome of 8.7, MonthlyIncome of 9760.7, NumCompaniesWorked of 2.9, TotalWorkingYears of 10.9 and YearsAtCompany of 10.6.
- **Cluster 1** - Contains 85 elements, it is characterized from values having a mean age of 32.5, DistanceFromHome of 9.4, MonthlyIncome of 18088, NumCompaniesWorked of 2.2, TotalWorkingYears of 10.4 and YearsAtCompany of 13.4.
- **Cluster 2** - Contains 259 elements, it is characterized from values having a mean age of 46.2, DistanceFromHome of 9.7, MonthlyIncome of 6407.1, NumCompaniesWorked of 2.5, TotalWorkingYears of 10.9 and YearsAtCompany of 6.7.
- **Cluster 3** - Contains 55 elements, it is characterized from values having a mean age of 36.8, DistanceFromHome of 9.7, MonthlyIncome of 6407.1, NumCompaniesWorked of 2.5, TotalWorkingYears of 10.9 and YearsAtCompany of 6.7.
- **Cluster 4** - Contains 151 elements, it is characterized from values having a mean age of 45.2, DistanceFromHome of 9.7, MonthlyIncome of 13290.6, NumCompaniesWorked of 3.4, TotalWorkingYears of 12.4 and YearsAtCompany of 11.3.
- **Cluster 5** - Contains 311 elements, it is characterized from values having a mean age of 36.4, DistanceFromHome of 8.6, MonthlyIncome of 4652, NumCompaniesWorked of 2.5, TotalWorkingYears of 11.1 and YearsAtCompany of 5.2.

2.2 Density-based clustering

In this section, we will describe the choices made while clustering with DBScan. We considered several subsets of attributes, and, for each of them, we analysed the algorithm's trend depending on the variations of epsilon and minpts. Then, we plotted the number of clusters obtained in a heatmap.

Distance and cluster numbers

We decided to consider only subsets of attributes, without categorical and discrete attributes having a small domain. Therefore, we chose numeric variables that were not correlated each other (redundant attributes were merged to improve efficiency: an example is YearsMean).

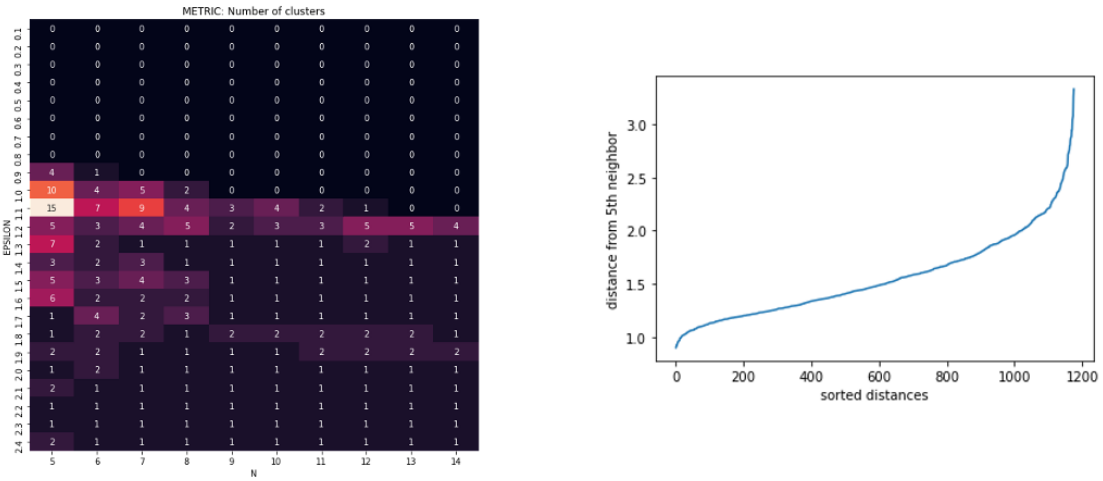


Figure 2.4: The heatmap with number of cluster and distance from 5-th neighbors

Since this type of algorithm doesn't care about the shape of the clusters, instead of using the silhouette index, like in K-Means, we decided to evaluate the different clusters using the heatmap. So, for each pair $< eps, minpts >$ it gives us the number of clusters found. The algorithm was performed for all the epsilon values of the interval [0.1,2.5] and mintpts [5.15].

Iperparameters chosen

Analysing the heatmap, we immediately realised that for low values of epsilon all points are classified as noise points (0 clusters) instead, for high values, we tend to obtain a single cluster. Our reasoning, looking at the heatmap led us to go deeper in our research for a solution able to provide at most 2-3 clusters. We divided the research in two areas, for epsilon $\in [1.1, 1.2]$ and N$\in [8, 11]$ and to epsilon $\in [1.8, 1.9]$ and N $\in [9, 14]$. Subsequently, from the distance function (5-NN distances) we can see that about 85% of the points have a distance ranging from 1.0 to 2.0; moreover with an epsilon greater than 2.0 the distances grow very fast and we will most likely include in the clusters even the noise points.

The distance graph leads us to choose the hyper-parameters from the second zone, not exceeding the epsilon limit. In fact, as we can see from the tables 2.5, most of the points in the first area are noise points, and the cluster identified contains a few relevant values. The second area confirms the presence of a single large cluster and a quite low number of noise (about 10%), so we will choose hyper-parameters belonging to this area. Any variation of the hyper-parameters increases noise and proportionally decreases points from this relevant cluster.

eps	minpts	noise points	value counts
1.1	8	1069	[79, 12, 7, 9]
1.1	9	1097	[27, 43, 9]
1.1	10	1125	[26, 8, 9, 8]
1.1	11	1143	[25, 8]
1.2	8	845	[290, 5, 25,7, 4]
1.2	9	892	[260, 24]
1.2	10	927	[179, 52, 18]
1.2	11	957	[162, 45, 12]

eps	minpts	noise points	value counts
1.8	9	183	[977, 16]
1.8	10	195	[968, 13]
1.8	11	209	[954, 13]
1.8	12	209	[945, 13]
1.9	9	128	[1048]
1.9	10	134	[1042]
1.9	11	158	[1003, 15]
1.9	12	166	[994, 16]

Figure 2.5: Core and noise point changing eps and minpts

We tested the algorithm using metrics different from the Euclidean one, like Minkowski, cosine and cityblock. Unfortunately the result remained the same: a big cluster is prevailing.

Even considering other subsets, the result doesn't change. The following scatter-plots describes the distribution as a function of pairs of attributes chosen from a subset with 7 attributes, with an epsilon = 1.8 and minpts = 11. In particular, the subset is composed by HourlyRate, DailyRate, MonthlyRate, DistanceFromHome, MonthlyIncome, TotalWorkingYears, Age and YearsAtCompany.

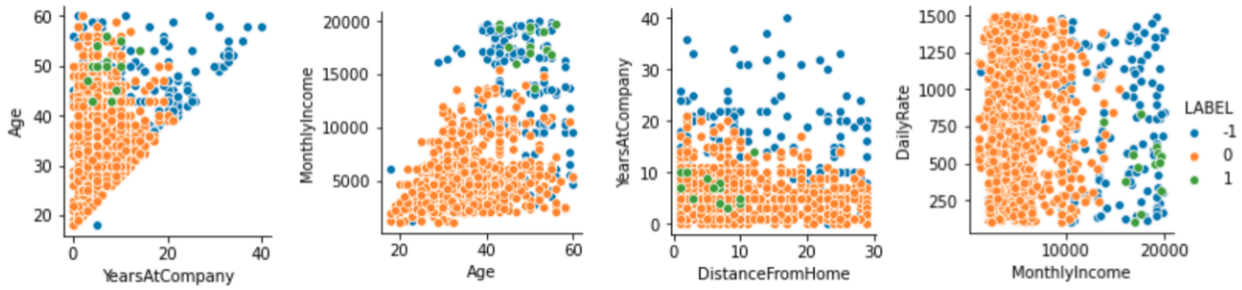


Figure 2.6: dbscan scatter

Considering the density parameter, it's clear to see that doesn't provide any further information on dataset's structure. Regardless of the possible subsets of attributes considered, this won't change. The algorithm identifies the points as a single large cluster.

2.3 Hierarchical Clustering

We considered the same subset of numerical attributes that we used for DBSCAN, taking into account all the records available inside the dataset. The dendrograms plotted show: into the x-axis all the values that have been grouped to ease the reading process and into the y-axis the distance between clusters. Depending on the methods used, hierarchical clustering gives better results compared to the density based approach.

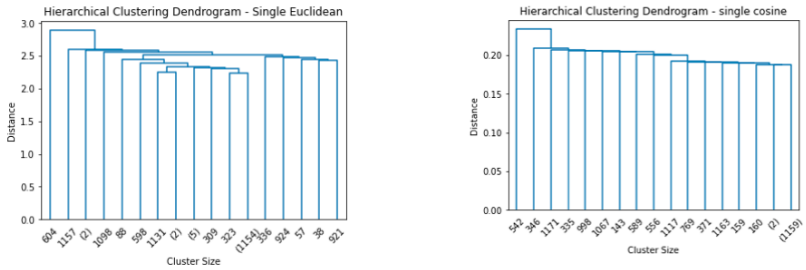


Figure 2.7: Dendrograms with single method, with euclidean and cosine metrics

As expected, the single-link method is the one that gave us the worst results because the clusters tend to come together too rapidly at a really small distance. This behaviour, as it was even confirmed by DBSCAN, is due to the continuity of the records. Both the metrics used (euclidean distance and cosine) are unsatisfactory.

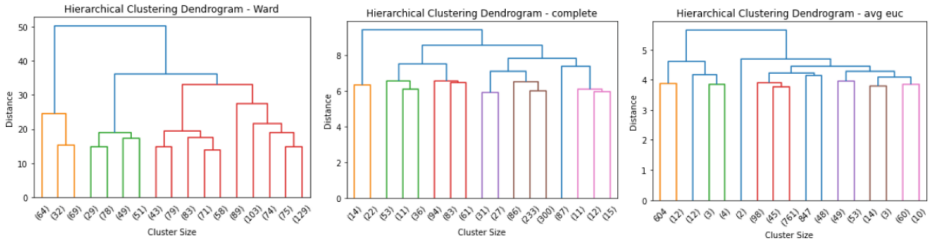


Figure 2.8: Dendrograms with Ward, complete and average method.

Differently from the previous one examinated, Ward, complete and average methods give us satisfactory results. The first one listed, which tends to minimize the variance inside clusters, splits the dataset into three clusters. The last two listed, i.e. average and complete method, seems to match up together, partitioning the dataset into 6 clusters of similar dimension; those two methods are even giving a similar result than using K-Means method.

2.4 Clustering Evaluation

K-Means managed to divide the dataset into 6 different clusters having a mean dimension of 196 and a standard deviation equal to 114. Taking into account the fact that the cluster has been divided into 6 clusters, having a silhouette score equal to 0.6489, this means that the clusters obtained are well divided, so we can say that the dataset points are really sparsed along their domain, having some parts way more dense than others. Doing an overall evaluation, we can say that K-Means is doing an excellent job clustering this dataset.

DBSCAN instead managed to divide the dataset into only one cluster, this is due to the fact that the input data is composed mostly from near and dense values. So it arises the fact that this kind of clustering algorithm doesn't really work well for this kind of dataset.

With hierarchical clustering we evaluated the computation done by this algorithm executed by using ward, complete and average methods by using silhouette score. From this analysis it came out that the best way to clusterize this dataset is by using the Ward method, because it gives the highest value of Silhouette score, compared to the other methods, which is 0.5094.

Clustering Technique	Clustering Quality	Silhouette Value	Number of cluster	Distance metric
K-Means	Excellent	0.6489	6	euclidean
DBScan	Poor	0.5859	1	euclidean
Hierarchical	Good	0.5094	3	euclidean

Figure 2.9: K-Means, DBSCAN and Hierarchiacal clustering algorithms comparision.

At the end, we can state that, between all the three clustering method analyzed, the worst is surely DBSCAN, regarding the fact that it scored a pretty high value of silhouette (0.5859), but this value is due to the fact that, as we described while speaking about the K-Means, the data points are really dense. K-Means

and Hierarchical clustering are the ones that gave a better result on clustering this dataset but, taking into account the silhouette scores, K-Means is definitely the one that does the best job.

3 Classification

Differently from the previous chapter, where we adopted a supervised method to understand datasets structure, now we're always using a supervised approach by which, starting with training data, we've build and train a model (using train set) capable to predict the Attrition level on new records (test set), trying to maximize the accuracy and to minimize generalization errors on new data, avoiding so overfitting phenomena.

3.1 Data Preparation and Procedure

Before starting to build the classificators, is very important to make some previous considerations regarding data cleaning, feature reshaping and data partitioning. First of all, the handling of missing values, outliers and dimensionality reduction follows the procedure already described into the Data Understanding chapter, so we used the same principles for the substitution of missing values and for treating the outliers. As we can see from the Correlation Matrix (fig 1.12), attributes have been considered removing Over18 and StandardHours for the reasons previously explained; we have also created YearsMean as the mean between the attributes YearsInCurrentRole, YearsSinceLastPromotion and YearsWithCurrManager and, lastly, considering MonthlyIncomeTrans, instead of MonthlyIncome, because its values were not well distributed. The algorithm will be executed on a dataset composed by 28 attributes. Once we have decided which variables to take into account we reshaped all the nominal features in way to be able to use the classification algorithm.

	Training Set	Test Set
Total Records	1176	294
Attrition = 'No'	984	249
Attrition = 'Yes'	192	45

Figure 3.1: Dataset Partitioning

The attrition dataset have been given us already divided into two datasets, one for training model and one for testing it. The test-set was used only for evaluating the selected model that was built using the training-set. We used those latter to build a collection of classifiers basing on iper-parameters obtained by using GridSearchCV and RandomizedSearchCV. Afterwards we evaluated the accuracy using k-fold cross validation, where k is the number of subsets in which training data have been divided. With k=4, we splitted the training set into 4 different folders, each of which, every time, will be used for the validation. By doing so, at the end of the iterative process, we have built 4 different models having 4 different evaluations each, using the same data.
If we would have deleted more records into the preprocessing step, we would have been able to join training and test data in way to use cross validation on the whole dataset. However, by keeping the integrity of the dataset we preferred to keep the test set separated in way to have 20% of the data not used in way to make an efficient evaluation. The classifiers realized are compared according to the f-measure. Consequently, our objective is to select the classifier that maximizes that value on the training set, evaluating the accuracy and the generalization error on new data

3.2 Decision Tree Classification

3.2.1 Basic Decision Tree

With the aim of realizing the best Decision Tree as possible, we firsly created a Simple Decision Tree (SDT), then we optimized it by generating an Optimized Decision Tree (ODT). At the end, we ran the classification algorithm with the parameters shown into the Figure 3.1 and using the GINI score to evaluate its purity.

max depth	min samples leaf	min samples split	Data Type	Accuracy	Precision	Recall	F-Score
2	1	2	Training (4-fold cv)	0.8512 (+/- 0.03)	0.8343 (+/- 0.03)	0.8512 (+/- 0.03)	0.8265 (+/- 0.06)
			Training	0.8494	0.8831	0.9451	0.9131
			Test	0.8231	0.8773	0.9196	0.8980

Figure 3.2: SDT classificator

The estimations shown into the table are calculated by evaluating SDT on training set (both using cross validation and the entire training set), and afterwards with the entire collection of the new records. The generated tree represents the basic classifier from which develop the optimizations that will be described in the next section.

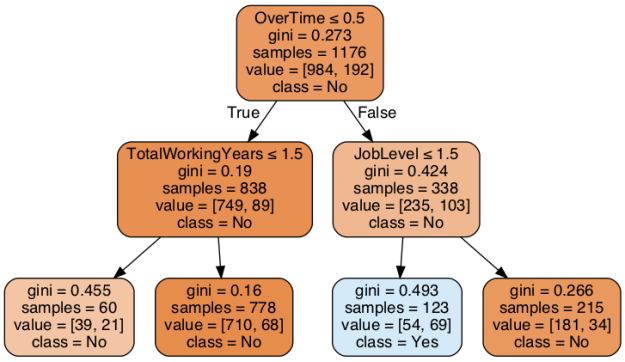


Figure 3.3: Simple Decision Tree with 2 levels

3.2.2 Optimized Decision Tree

The tuning of the iper-parameters have been done with the objective of increasing the complexity of the base SDT model in way to find the classificator that generalizes better, evaluating it basing on the f-score of the training data. We used GridSearch and RandomizedSearch algorithms comparing them with the parameters shown into the table and measuring the purity of the nodes to split using GINI score, despite entropy manages to highlight better pure nodes, as the function increases/decreases faster for values on the extremes of the domain.

Algorithm	Parameters	
Grid Search	max_depth	[2:10]
	min_samples_leaf	[2, 5, 10, 20, 30, 40, 50, 100]
	min_samples_split	[1, 5, 10, 20, 30, 40, 50, 100]
Randomized Search	max_depth	[2:100]
	min_samples_leaf	[2, 5, 10, 20, 30, 40, 50, 100, 150, 200]
	min_samples_split	[1, 5, 10, 20, 30, 40, 50, 100, 150, 200]

Figure 3.4: ODT parameters configuration

The following table shows some classifiers obtained configuring the precedent algorithms with the parameters just shown. Particularly, using GridSearch we obtained the first group of classifiers, the remaining ones were obtained using RandomizedSearch. For each model we calculated the accuracy, precision, recall and f-score both on the training-set (both etirely and both 4-fold cv) and on the test-set. We didn't reported classifiers with a maximum depth greater than 50 because our objective is to find a model that minimizes the generalization error on new data and, in the meanwhile, aims to minimize the complexity. If depth would be greater than 50 it could lead to overfitting.

Analyzing the values, we selected the model with the f-measure higher on the training-set, at parity of this value we will consider the roc-auc, accuracy, recall and precision values respectively. Particularly, the first group of models, where the tuning was executed using GridSearch, the f-score value is 0.8215 with 0.3 as deviation on cross validation and 0.9258 on the entire training set. It is the best classifier between the ones we found, even if the other ones had really little differences between each other. Keeping in mind these considerations, we selected the first model (with minimum complexity) whose evaluation on the test set was

Max depth	Min simples leaf	Min samples split	Data Type	Accuracy	Precision	Recall	F-Score
5 6 7 8 9	20 20 20 20 20	[2, 5, 10, 20, 30, 40] [2, 5, 10, 20, 30, 40] [2, 5, 10, 20, 30, 40] [2, 5, 10, 20, 30, 40] [2, 5, 10, 20, 30, 40]	Training (4-fold cv)	0.8401 (+/- 0.00)	0.8186 (+/- 0.03)	0.8401 (+/- 0.00)	0.8215 (+/- 0.03)
			Training	0.8707	0.8902	0.9644	0.9258
			Test	0.8503	0.8897	0.9397	0.9140
2 2	5 40	150 10	Training (4-fold cv)	0.8495 (+/- 0.03)	0.8020 (+/- 0.12)	0.8495 (+/- 0.03)	0.8212 (+/- 0.07)
			Training	0.8494	0.8831	0.9451	0.9131
			Test	0.8231	0.8773	0.9196	0.8980
11 14 15 39	50 50 50 50	30 50 50 20	Training (4-fold cv)	0.8469 (+/- 0.03)	0.8246 (+/- 0.05)	0.8469 (+/- 0.03)	0.8306 (+/- 0.05)
			Training	0.8545	0.8613	0.9847	0.9189
			Test	0.8401	0.8633	0.9638	0.9108

Figure 3.5: ODT classificators

coherent with the validation effected on the model selection, having an f-score equal to 0.9140, higher than the other classifiers. Even accuracy and precision are the better ones.

ODT(1): max depth = 5, min samples leaf = 20, min samples split = 2 and test f-score of 0.9140.

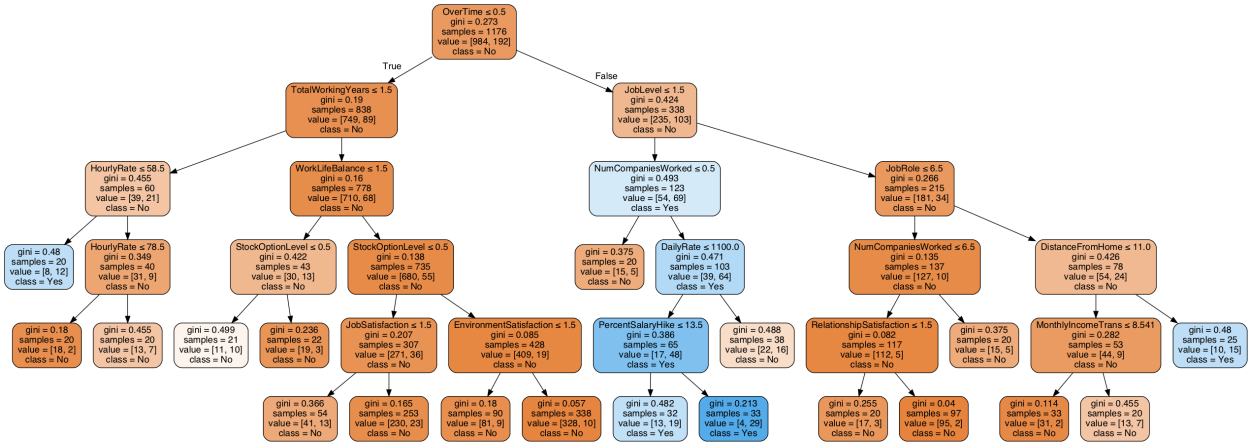


Figure 3.6: ODT classificator with max depth=5, min simples leaf=20 e min samples split=2

A valid alternative, even if losing 1.6% of f-score and 2.7% of accuracy, would be to select the group of classifiers with maximum depth equal to 2. To keep as low as possible the complexity of the classifier (pruning) is a good compromise. Important to highlight the similarity with SDT.

ODT(2): max depth = 2, min samples leaf = 40, min samples split = 10 and test f-score of 0.8980.

The representation of the tree is the same as SDT, decribed in the previous section (fig 3.3).

3.2.3 Random Forest Classification

We executed Random Forrest classification configuring the parameters as described on the table. The best estimator is represented by a very complex model, with depth equal to 46.

Algorithm	Parameters	
Random Decision Forest	max_depth	[2:200]
	min_samples_leaf	[2, 5, 10, 20, 30, 40, 50, 100, 150]
	min_samples_split	[1, 5, 10, 15, 20, 30, 50, 100, 150]
	criterion	['gini', 'entropy']

Figure 3.7: Random Forest parameters configuration

The model obtained, as expected, is not supportive. The improvement of the f-score test, equal to 0.0036, is not signicative if compared to an increment on the maximum depth of 41 levels.

max depth	min samples leaf	min samples split	Data Type	Accuracy	Precision	Recall	F-Score
46	1	10	Training (4-fold cv)	0.8512 (+/- 0.01)	0.8520 (+/- 0.04)	0.8529 (+/- 0.02)	0.7984 (+/- 0.02)
			Training	0.9404	0.9335	1.0	0.9656
			Test	0.8503	0.8596	0.9839	0.9176

Figure 3.8: Random Forest classificator

3.3 K-NN Classification

K-Nearest Neighbor classifies each instance basing on the majority class on the k nearest points. The proximity is calculated using the euclidean distance. This makes the calculation to be expensive in the case that the dataset is composed by a few attributes. As we can read from the table. we executed the algorithm with a maximum k value equal to 12, obtaining so different evaluation according to the number of the neighbors considered.

K	Data Type	Accuracy	Precision	Recall	F-Score
1	Training (4-fold cv)	0.7355 (+/- 0.05)	0.7407 (+/- 0.04)	0.7355 (+/- 0.05)	0.7378 (+/- 0.04)
	Test	0.7619	0.8565	0.8634	0.86
2	Training (4-fold cv)	0.8189 (+/- 0.02)	0.7404 (+/- 0.06)	0.8189 (+/- 0.02)	0.7629 (+/- 0.02)
	Test	0.8299	0.8515	0.9678	0.9060
3	Training (4-fold cv)	0.7951 (+/- 0.04)	0.7457 (+/- 0.04)	0.7951 (+/- 0.04)	0.7624 (+/- 0.03)
	Test	0.8095	0.8534	0.9357	0.8927
4	Training (4-fold cv)	0.8240 (+/- 0.02)	0.7338 (+/- 0.08)	0.8240 (+/- 0.02)	0.7604 (+/- 0.02)
	Test	0.8299	0.8515	0.9678	0.9060
5	Training (4-fold cv)	0.8129 (+/- 0.04)	0.7350 (+/- 0.04)	0.8129 (+/- 0.04)	0.7606 (+/- 0.03)
	Test	0.8333	0.8597	0.9598	0.9070
6	Training (4-fold cv)	0.8342 (+/- 0.01)	0.7503 (+/- 0.07)	0.8342 (+/- 0.01)	0.7684 (+/- 0.01)
	Test	0.8469	0.8517	0.9919	0.9165

K	Data Type	Accuracy	Precision	Recall	F-Score
7	Training (4-fold cv)	0.8299 (+/- 0.02)	0.7814 (+/- 0.11)	0.8299 (+/- 0.02)	0.7675 (+/- 0.01)
	Test	0.8367	0.8501	0.9799	0.9104
8	Training (4-fold cv)	0.8333 (+/- 0.01)	0.6997 (+/- 0.00)	0.8333 (+/- 0.01)	0.7607 (+/- 0.00)
	Test	0.8469	0.8493	0.9959	0.9168
9	Training (4-fold cv)	0.8316 (+/- 0.01)	0.7514 (+/- 0.14)	0.8316 (+/- 0.01)	0.7629 (+/- 0.01)
	Test	0.8435	0.8487	0.9919	0.9148
10	Training (4-fold cv)	0.8350 (+/- 0.01)	0.6999 (+/- 0.00)	0.8350 (+/- 0.01)	0.7615 (+/- 0.00)
	Test	0.8469	0.8493	0.9959	0.9168
11	Training (4-fold cv)	0.8350 (+/- 0.01)	0.6999 (+/- 0.00)	0.8350 (+/- 0.01)	0.7615 (+/- 0.00)
	Test	0.8469	0.8493	0.9959	0.9168
12	Training (4-fold cv)	0.8359 (+/- 0.00)	0.7000 (+/- 0.00)	0.8359 (+/- 0.00)	0.7619 (+/- 0.00)
	Test	0.8469	0.8469	1.0	0.9171

Figure 3.9: K-NN classificator. Confusion matrix and roc area for k=5

From a quick analysis of the results we observed that the f-score on the training data goes from a minimum of 0.7378 with k = 1 to a maximum of 0.7684, obtained with k = 6. In fact, we choose the model with k = 6 whose test f-score = 0.9165. In general the variations of the f-score on new records are not significative, they vary limitedly from the 90% to the 91% (excluding k = 1), particularly we have 0.8927 with k = 2 and 0.9171 with k = 12. For values of k ¿ 12 the f-score remains the same, therefore we didn't kept on iterating for looking for a better estimate.

KNN roc-auc	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10	K=11	K=12
	0.5172	0.5234	0.5172	0.5465	0.5182	0.5121	0.5091	0.5070	0.5091	0.5091	0.5

Figure 3.10: Area under the curve for every k value.

Considering equally the all f-score values (with k = 1 excluded), we looked for the k having the gretest roc-auc (area under the curve) value. The table shows us that even this estimate don't suffer from evident alterations, therefore for k = 5 we have a better auc with respect to the others, equal to 0.55. So we choose that k (= 5) for building (learning) the model with K-NN. In this situation, the choice of a k, rather than another, doesn't lead to substantial differences as all the estimates are really close.

3.4 Discussion of the best prediction model

Before describing which, between the classifiers found, predicts better new data, it is important to make some clarifications. In this context is reasonable to not consider a model better with respect to another only if the f-score is greater or not. It is important to evaluate estimates all as an entire, keeping in consideration even accuracy, precision, recall and considering the area under the roc curve (roc-auc). We need to keep in count even the complexity of the classifiers, according to the maximum depth of the tree that was used to build (learn) it. Another important factor is the computational cost for realizing the model: K-NN in fact, for each record has to calculate the euclidean distance for each point belonging to a space that, in our case, is of dimension 28.

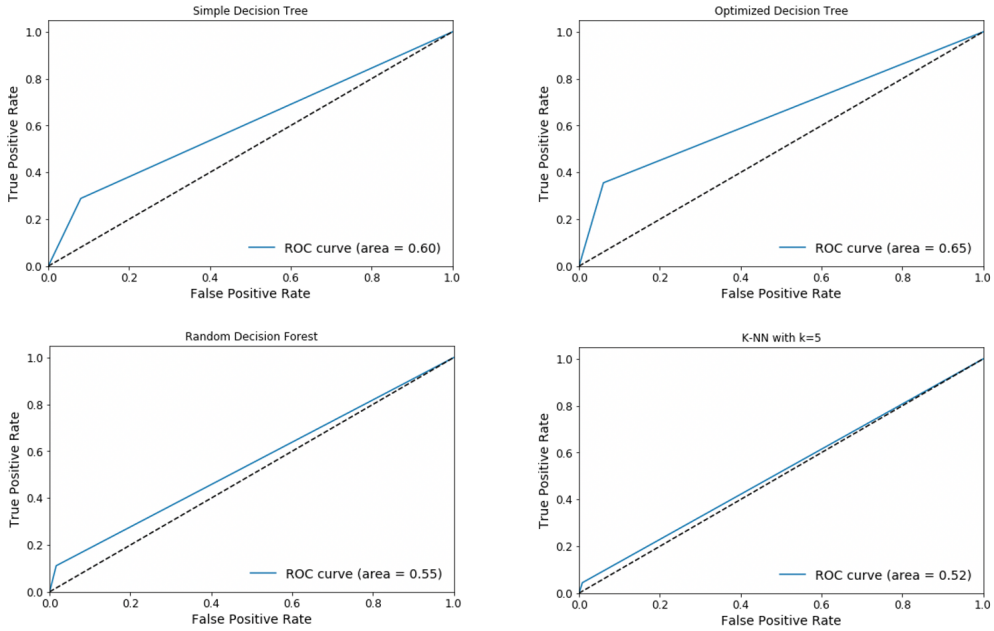


Figure 3.11: Area under the rocs curves for each model

At a first overview we can affirm that, independently from the examined classifier, the estimates don't suffer from big variations (apart from recall in one case). Particularly, the accuracy vary less than 2.72%, precision of 3.01%, recall of 6.43% and f-score of 1.96%. Consequently, excessively complex classificators have been avoided, like the one obtained using Random Forest that, even if it had higher values of f-score and recall, it had a maximum depth of 46 that, in this particular case, doesn't overfit, but it could on new future data: for example more records related to new employees with different attitudes and habits.

Model	Accuracy	Precision	Recall	F-Score
Simple Decision Tree	82.31%	87.73%	91.96%	89.80%
Optimized Decision Tree	85.03%	88.97%	93.97%	91.40%
Random Forest Model	85.03%	85.96%	98.39%	91.76%
K-Nearest Neighbors Model with k=5	83.33%	85.97%	95.98%	90.70%

Figure 3.12: Models Evaluation on test set.

Subsequently, it is interesting to compare the SDT model with the one realized using KNN, with k = 5. This last one has the advantages of being simple to build and the absence of training steps. Furthermore, in our case it gives good estimates not only by using a value of k equal to 5. The cons of this approach is on the fact that its execution is really slow and that it is really computationally expensive due to the calculations of the euclidean distances basing on the dimension of the dataset (curse of dimensionality). On the other side, SDT has a max depth = 2, which value leads on having a computational complexity really low, the estimates

are as good as in KNN, but with a discrete advantage on the roc-auc side (fig 3.11). Those considerations lead us to prefer the SDT approach.

The classificator that according to our point of view has the best predicting potential compared to the others is ODT: which is an evolution of SDT. The model has a low complexity because the tree has a maximum depth of 5. Furthermore it has the maximum value of roc-auc equal to 0.65 and has higher value on accuracy and f-score, respectively 85.03% and 91.40%.

At the end, we can imagine a preference order as it follows: ODT, SDT, K-NN with k = 5 and Random Forest Model.

4 Association Rules and Pattern Mining

4.1 Pattern Mining

In this section we are going to identify the rules that describe specific patterns in the data and to discover hidden information related to the dataset.

4.1.1 Attribute selection and binning

At first, we decided to remove the fields of the dataset that wasn't providing us any useful information, those attributes were *Over18* and *StandardHours*. We then proceeded filling the missing values in the same way as we did into the Data Understanding chapter. We decided then to remove the redundant attributes like DailyRate, MonthlyRate, etc. because they'd lead us to have association rules based on values indicating the exact same values. At the end, the fields that we removed (or adjusted) are: *DailyRate*, *MonthlyRate*, *HourlyRate*, *PercentSalaryHike*, *MonthlyIncome*, *YearsInCurrentRole*, *YearsSinceLastPromotion*, *YearsWithCurrManager*, *Over18* and *StandardHours*.

To perform the Pattern minig task we needed to discretize all the numerical attributes present in the dataset, for doing so we divided all those attributes in quartiles, in way to be able to be less restrictive while looking for association rules.

4.1.2 Frequent item-set extraction

For extracting frequent item-sets we ran the Apriori algorithm based on the attributes described before. After trying with different values of support, we decided that the best support value would have been 0.3. We noticed that the number of item-sets found changes significantly by varying the value of the support given in input to the algorithm.

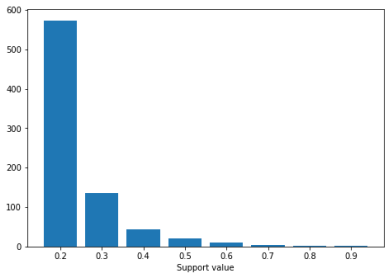


Figure 4.1: Support values bar chart

Patterns	Support
Male	0.61479
No	0.91241
PerformanceRating:3.0	0.86564
Research & Development	0.65391
Travel_Rarely	0.74064
WorkLifeBalance:3	0.61054
No, PerformanceRating:3.0	0.79251
No, Research & Development	0.60204
No, Travel_Rarely	0.67431
PerformanceRating:3.0, Travel_Rarely	0.64370

Figure 4.2: Most frequent itemsets

From the Figure 4.1 represented above, we can see how the number of item-sets changes in relation to the variation of the support value. By looking at it, it is clear how a huge number of item-sets were found using a support value equal to 0.2; we decided then that the best support value would be 0.3 because it'd let us to have a discrete amount of item-sets to use to find association rules. From the table 4.2, we can see the most popular item-sets on the entire dataset, as measured by the proportion of transactions in which an item-set appears.

4.1.3 Association Rules

In this section we will describe the most interesting association rules found using the Apriori algorithm. To help us decide which would have been the best value of confidence and lift to use, we decided to draw a scatter plot, in way to see where we would be able to find the best association rules.

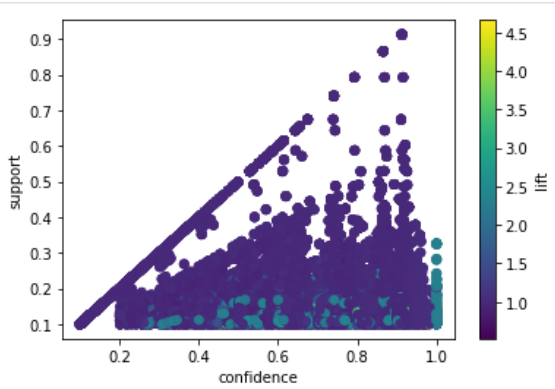


Figure 4.3: Scatter plot with support, confidence and lift values

Considering the graph in Figure 4.3, we decided that the best values for support confidence and lift would have been, respectively, 0.3, 0.6 and 1.5. It is clear from the table that there were records having a high value of lift (between 2 and 3) for itemsets having a confidence lower than 0.6 and a support less than 0.3, but those values were relative to connections relative, for example, between people working into the Sales department, working as Sales Executive, or for people working in the Research & Development department, working as Research Scientist, so they were indeed useless connections.

Patterns	Confidence	Lift
JobLevel:1, TotalWorkingYears:(-0.001, 6.0]	0.62331	2.01934
NumCompaniesWorked:(-0.001, 1.0], TotalWorkingYears:(-0.001, 6.0]	0.79063	1.58666
Male, Single, StockOptionLevel:0	0.64229	2.31700
Married, No,PerformanceRating:3.0, StockOptionLevel:1	0.64864	1.51351
Married, StockOptionLevel:1, Travel_Rarely	0.69186	1.50115
No, NumCompaniesWorked:(-0.001, 1.0], TotalWorkingYears:(-0.001, 6.0]	0.68044	1.50696
No, PerformanceRating:3.0, Single, StockOptionLevel:0	0.74412	2.26707
NumCompaniesWorked:(-0.001, 1.0], PerformanceRating:3.0, TotalWorkingYears:(-0.001, 6.0]	0.68870	1.59119
PerformanceRating:3.0, Single, StockOptionLevel:0	0.86161	2.27699
Research & Development, Single, StockOptionLevel:0	0.64490	2.32641

Figure 4.4: Most relevant Association rules

Represented on the Figure 4.4 we have the most meaningful association rules. It is clear that, as it would be expected, the most related rules occure between JobLevel, TotalWorkingYears and NumCompaniesWorked, when those values belong to the lower quartile.

4.1.4 Association Rules for missing values replacement

After having found the Associations Rules present on the dataset, we decided to replace the missing values of the attribute PerformanceRating by following the steps described above:

1. For doing it we used the Apriori algorithm with a values of support and confidence equal to 0.4 and 0.6.

2. Among all the association rules found, we used only the ones containing the element PerformanceRating, and used it to fill the empty values.

4.1.5 Association Rules for predicting the target variable

For predicting the target variable using the association rules discovered and discussed before we followed those steps:

1. Select the rules with higher confidence and lift values.

2. Use those rules to predict the target variable, by looking for the association rules found into the attributes to predict.

3. Plot a confusion matrix in order to see whether the results are satisfactory or not, even comparing them with the results obtained by using classification algorithms.

		Predicted Values	
		Yes	No
Actual Values	Yes	8	184
	No	40	943

Figure 4.5: Confusion matrix

As we can see from the confusion matrix represented on the Figure 4.5, we have a 83,3% of possibilities of finding a false positive value and there is the 19.5% of possibilities of finding a false negative value.