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

Chapter 1

Data Understanding

1.1 Data Semantics

The Dataset contains a collection of working and personal information regarding 1470 different employes 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 Informations** - 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 informations 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

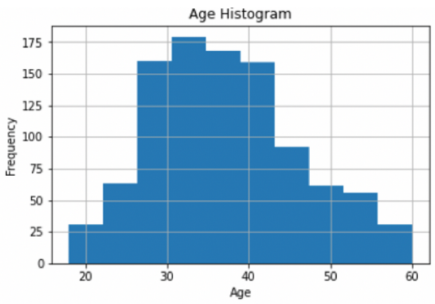
1.2.1 Numeric Attributes

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

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



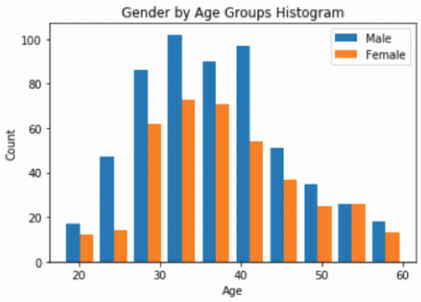


Figure 1.1: Age stats and histograms. On the right we also provides the number of attrited employees.

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).

	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

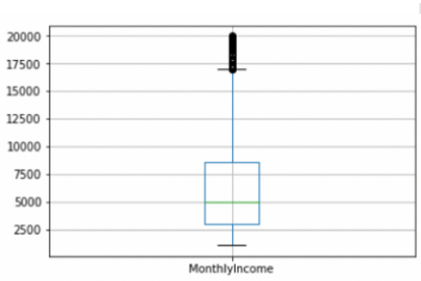


Figure 1.2: Salary amount group stats and MonthlyIncome boxplot in which appears outliers.

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

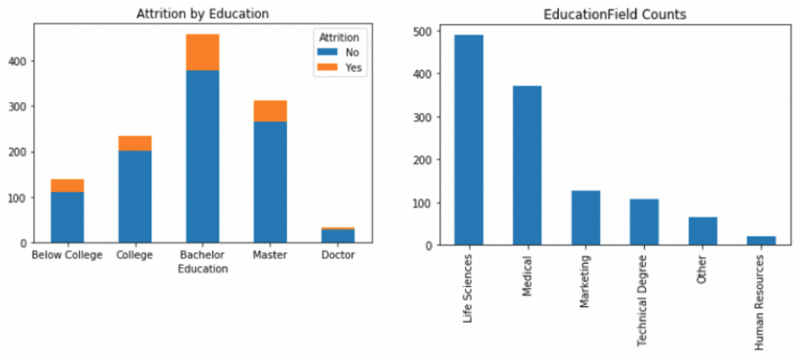


Figure 1.4: Education stats and bar charts.

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

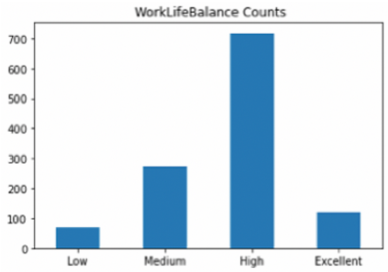


Figure 1.5: Attrition Key statistics and bar chart of WorkBalance.

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

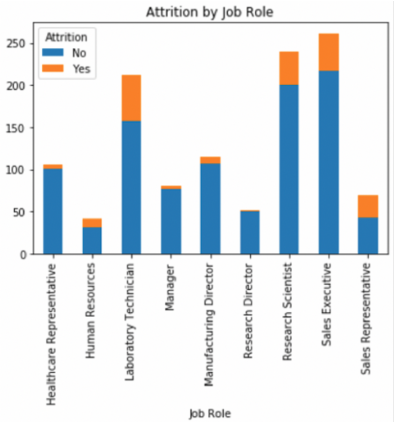


Figure 1.6: JobRole and JobLevel tables.

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 nor-

malisation, 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.

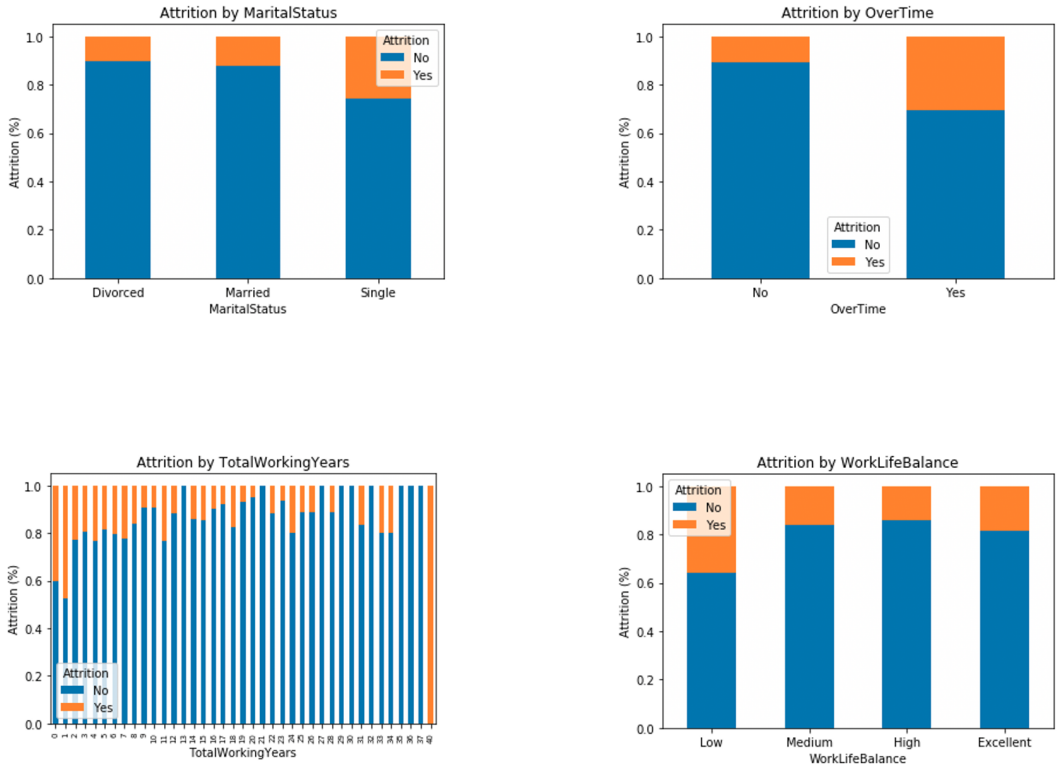


Figure 1.7: Attrition cross plots. In those plots the distributions are normalized. Therefore we can see the attrition frequence that an employee (among the total detected) has diversifying them basing on different values (values name listed on x-axis).

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 accord-

ing 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.

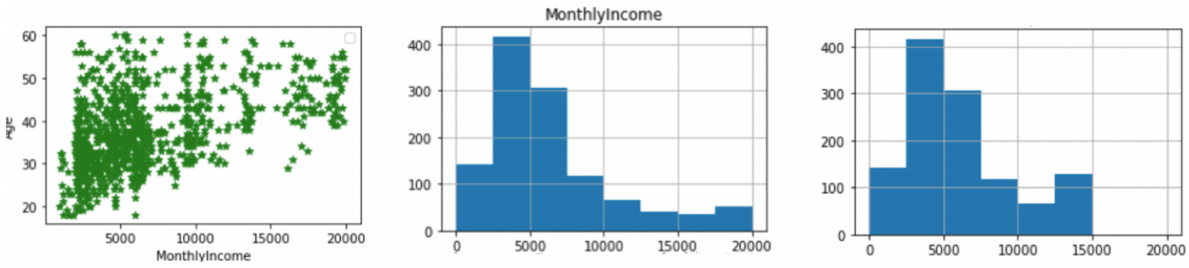


Figure 1.10: On the left the MonthlyIncome scatter shows a continuous distribution that leads to think that the outliers are not noise. The 2 histograms display as before and after replacing outliers with $Q3 + 1.5 * IQR$ could alterate wrongly the distribution.

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.
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

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

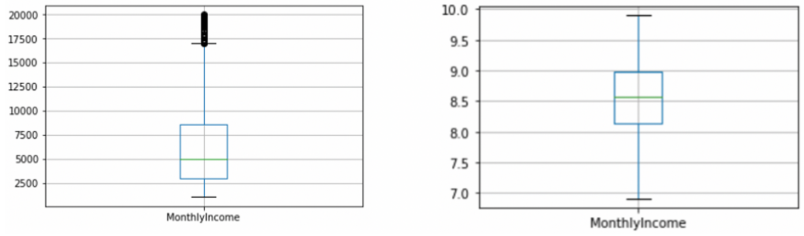


Figure 1.11: MonthlyIncome before and after logaritmic transformation.

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.

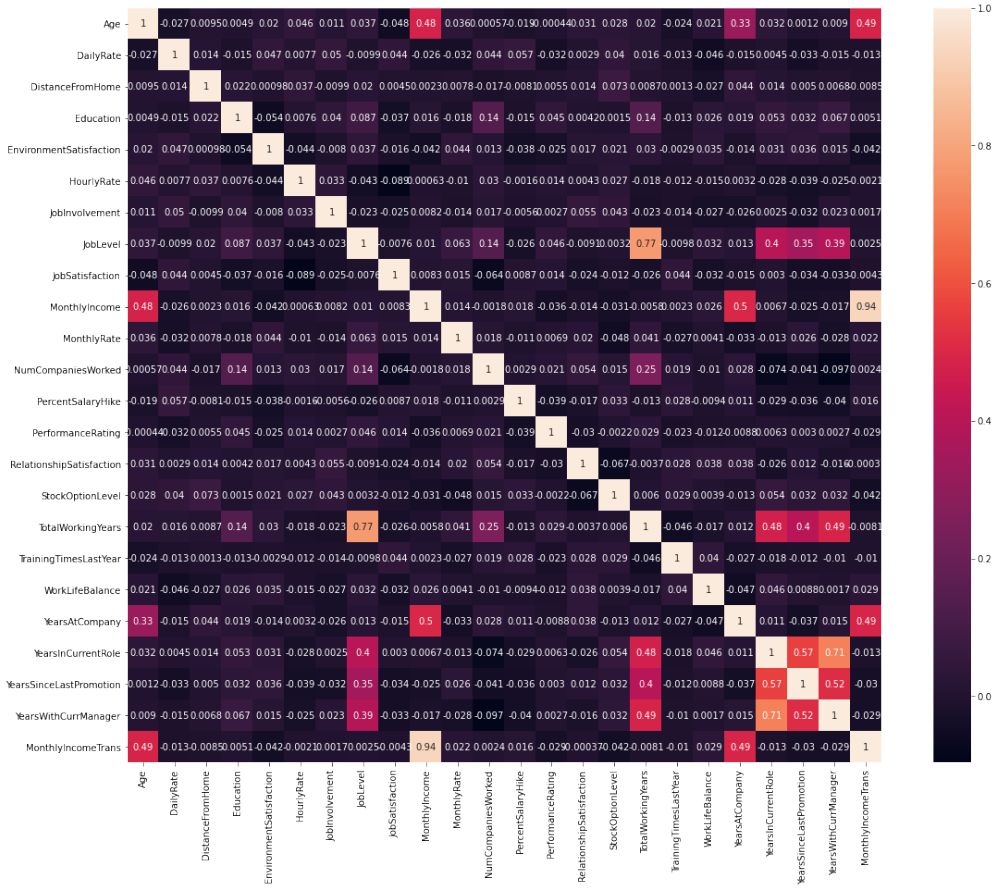


Figure 1.12: Correlation matrix. The gradient indicate more correlated attributes.

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.

Chapter 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.

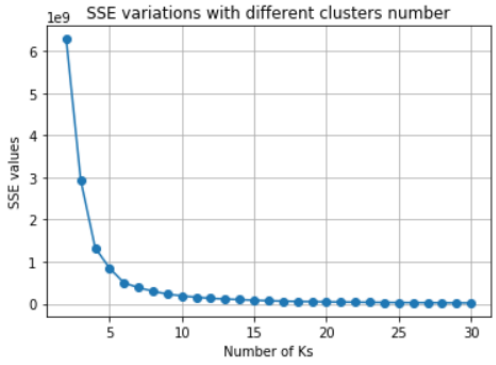


Figure 2.1: SSE plot for $k \in [2, 30]$

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.

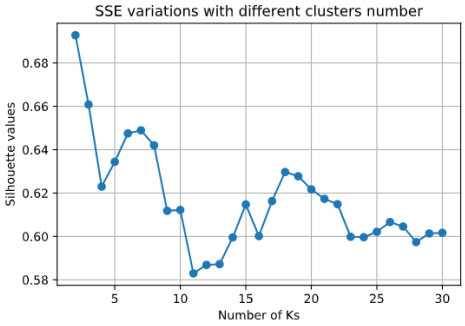


Figure 2.2: Silhouette plot for $k \in [2, 10]$

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.

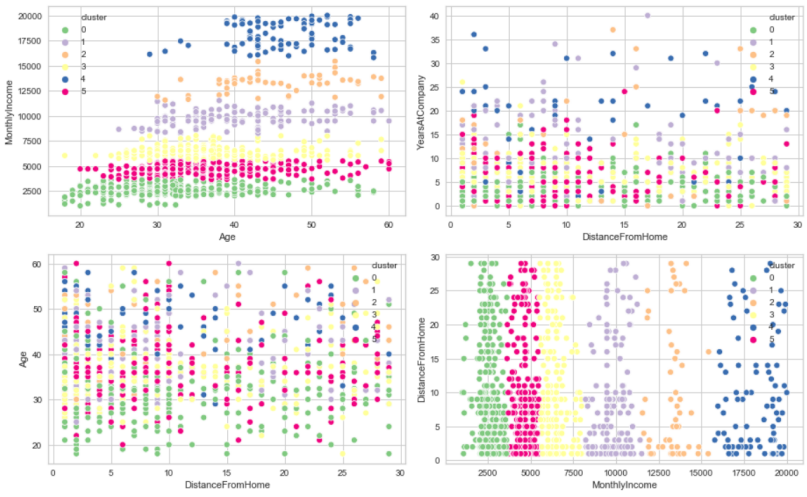


Figure 2.3: Scatter plot for different attributes.

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 mintpts. 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).

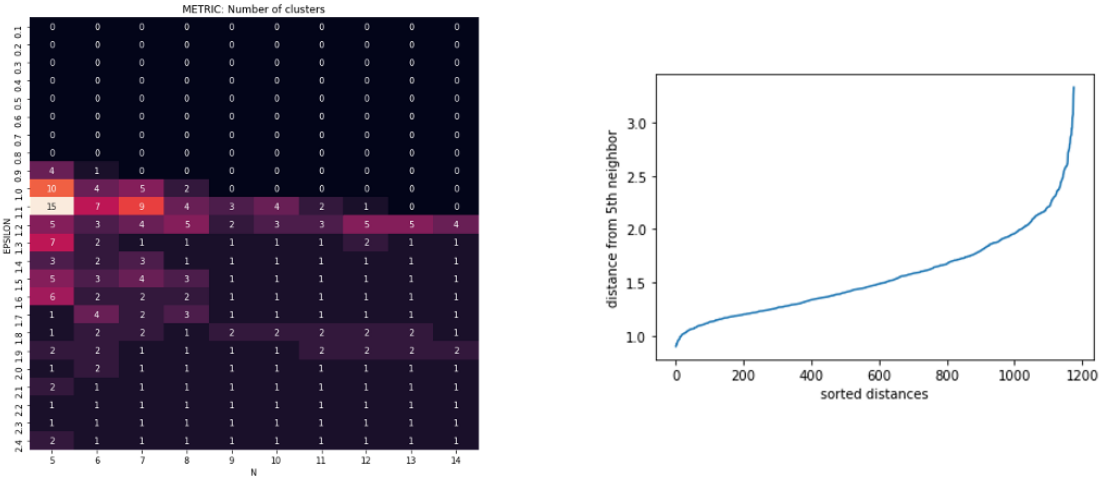


Figure 2.4: The heatmap with number of cluster and distance from 5-th neighbors

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 hyperparameters 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 hyperparameters belonging to this area. Any variation of the hyperparameters 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 city-block. 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.

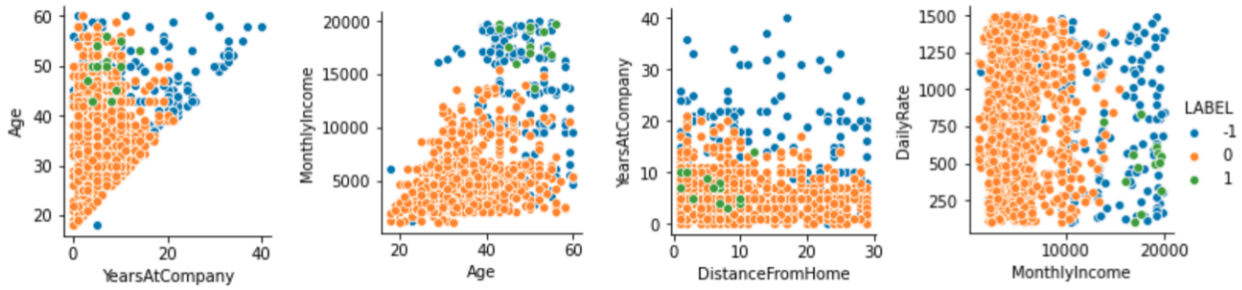


Figure 2.6: dbscan scatter

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.
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 DB-

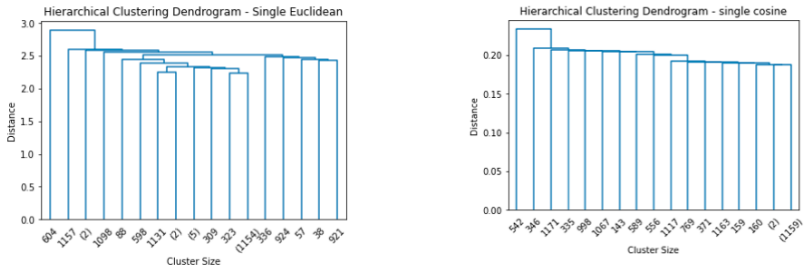


Figure 2.7: Dendrograms with single method, with euclidean and cosine metrics

SCAN, is due to the continuity of the records. Both the metrics used (euclidean distance and cosine) are unsatisfactory.

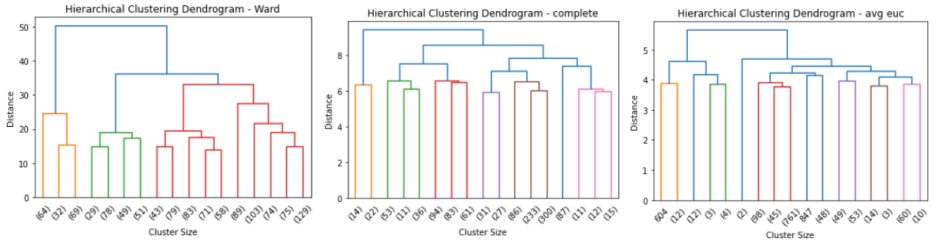


Figure 2.8: Dendrograms with Ward, complete and average method.

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.

Chapter 3

Classification

A differenza del precedente capitolo nel quale abbiamo adottato un metodo non supervisionato per capire la struttura del dataset, utilizziamo adesso un approccio supervisionato con il quale, a partire dai dati di training, allenare e costruire un modello in grado di predire l'attrition level su nuovi record (test set), massimizzandone l'accuratezza e cercando di minimizzare l'errore di generalizzazione su nuovi dati evitando il fenomeno dell'overfitting.

3.1 Data Preparation and Procedure

Prima di costruire i classificatori è fondamentale fare alcune considerazioni preliminari riguardo il data cleaning, il reshaping degli attributi (feature reshaping) ed il partizionamento del dataset (data partitioning). Innanzi tutto la gestione dei missing values, outliers e la riduzione della dimensione del dataset(handle missing values, outliers and dimensionality reduction) segue la linea già descritta nel capitolo sul data understinding, quindi adottando gli stessi principi per la sostituzione dei valori mancanti e il trattamento degli outliers. Come possiamo notare nella matrice di correlazione (fig.1.12), gli attributi sono considerati "al netto :)" di Over18 e StandardHours per le motivazioni precedentemente descritte. Inoltre, abbiamo creato YearsMean come media degli attributi YearsInCurrentRole, YearsSinceLastPromotion and YearsWithCurrManager, eliminandoli in quanto correlati e quindi ridondanti tra loro e, infine, considerando MonthlyIncomeTrans e non MonthlyIncome in quanto non ben distribuito. L'algoritmo viene quindi eseguito su un dataset composto da 28 attributi. Una volta stabilite quali variabili prendere in considerazione abbiamo effettuato il reshaping di tutte le features nominali in modo da poter utilizzare l'algoritmo di classificazione.

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

Figure 3.1: Dataset Partitioning

Il dataset attrition ci è stato fornito già ripartito in due dataset, uno per i dati di training ed uno per il test set. E' fondamentale sottolineare il fatto che il test set è stato utilizzato solo per valutare il modello selezionato e costruito sui dati di training. Abbiamo impiegato quest'ultimi per costruire un insieme di classificatori sulla base degli iperparametri ottenuti con GridSearchCV e RandomizedSearchCV. Successivamente ne abbiamo valutato l'accuratezza utilizzando k-fold cross validation, dove k è il numero di subsets in cui sono stati ripartiti i dati di training. Con k=4, abbiamo splittato il training set in 4 differenti folders, una delle quali, ogni volta, viene utilizzata per la validazione. Così facendo, al termine del processo iterativo, abbiamo 4 differenti modelli con 4 valutazioni differenti utilizzando gli stessi dati.
Nel caso in cui avessimo eliminato ulteriori record nel preprocessing, avremmo potuto unire i dati di training e di test e utilizzare la cross validation sull'intero dataset. Tuttavia, avendo mantenuto l'integrità dell'insieme di dati abbiamo preferito mantenere il test set separato così da avere un 20% di record effettivamente nuovi su cui effettuare una valutazione efficiente. I classificatori realizzati sono confrontati in funzione della f-measure. Di conseguenza, il nostro obiettivo è quello di selezionare il classificatore che massimizza tale valore nel training set, valutando l'accuratezza e l'errore di generalizzazione sui nuovi dati.

3.2 Decision Tree Classification

3.2.1 Basic Decision Tree

Con l'obiettivo di realizzare (learn) il miglior possibile Decision Tree, abbiamo creato prima un Simple Decision Tree (SDT), successivamente ottimizzandolo generando un Optimized Decision Tree (ODT). Di conseguenza, abbiamo eseguito l'algoritmo di classificazione con i parametri riportati in tabella e criterio di valutazione della purezza gini.

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

Le stime riportate in tabella sono state calcolate valutando SDT sul training set (sia cross validation che con l'intero training set) e successivamente con tutti i nuovi record. L'albero generato rappresenta il classificatore basilare dal quale sviluppare le ottimizzazioni descritte nella successiva sezione.

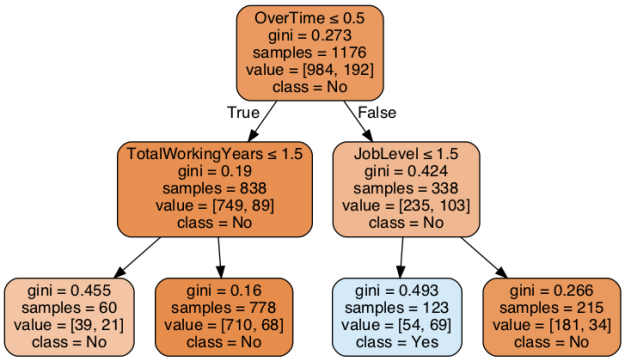


Figure 3.3: Simple Decision Tree with 2 levels

3.2.2 Optimized Decision Tree

Il tuning degli iperparametri è stato effettuato con l'obiettivo di aumentare la complessità del modello base SDT in modo tale da trovare il classificatore che meglio generalizza, valutandolo in base alla f-score sui dati di training. Abbiamo utilizzato Grid search e Randomized search algorithms configurandoli con i parametri riportati in tabella e misurato la purezza dei nodi da splittare con il criterio gini, nonostante l'entropia riesca a mettere più in risalto i nodi puri, in quanto la funzione cresce/descresce più rapidamente per valori agli estreim del dominio.

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

La tabella che segue mostra alcuni classificatori ottenuti configurando i precedenti algoritmi con i parametri appena riportati. In particolare il primo gruppo di classificatori lo abbiamo ottenuto con Grid Search, mentre i restanti con Randomized Search. Per ogni modello, abbiamo calcolato accuratezza, precisione, recall e f-score sia sui dati di training (interamente e tramite 4-fold cv) che sul test set. Non abbiamo riportato classificatori con profondità massima superiore a 50, in quanto il nostro obiettivo è cercare un modello che minimizza l'errore di generalizzazione sui nuovi dati e allo stesso tempo alla minor complessità. Una profondità superiore a 50 potrebbe tendere a generare overfitting.

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

Figure 3.5: ODT classificators

Analizzando i valori abbiamo selezionato il modello (model selection) con f-measure maggiore sul training set, a parità di essa considereremo i valori di roc-auc, accuratezza, recall e precisione. In particolare il primo gruppo di modelli, il cui tuning è stato eseguito con Grid Search, ha un f-score di 0.8215 con deviazione di 0.3 su cross validation e 0.9258 sull'intero training set. Rappresenta il miglior classificatore tra quelli rilevati, anche se gli altri modelli si discostano di molto poco (l'ultimo gruppo è quasi identico). Alla luce di queste considerazioni abbiamo selezionato il primo modello (ovviamente con complessità minima) la cui valutazione sul test set è coerente con la validazione effettuata nella model selection, avendo un f-score pari a 0.9140, superiore agli altri classificatori. Anche l'accuratezza e la precisione risultano le migliori.
ODT(1): max depth = 5, min samples leaf = 20, min samples split = 2 and test f-score of 0.9140.

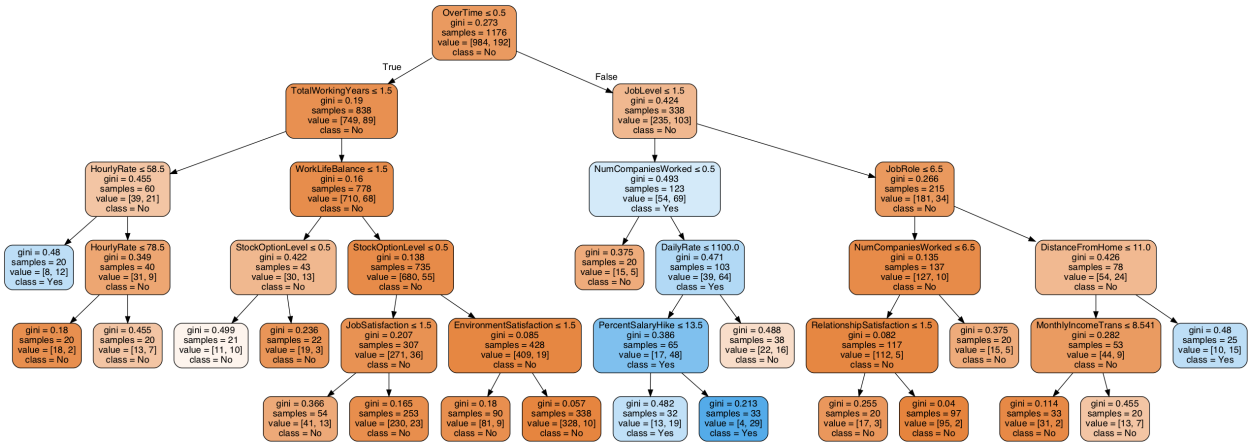


Figure 3.6: ODT classificator with max depth=5, min simples leaf=20 e min samples split=2

Una valida alternativa, a discapito dell'1.6% di f-score e dell'2.7% di accuratezza sarebbe quella di scegliere il gruppo di classificatori con profondità massima pari a 2. L'abbattimento al minimo della complessità del classificatore (pruned) è un ottimo compromesso. Da evidenziare la similitudine con SDT.
ODT(2): max depth = 2, min samples leaf = 40, min samples split = 10 and test f-score of 0.8980.
La rappresentazione dell'albero è identica a SDT, descritto nella precedente sezione (fig.3.3).

3.2.3 Random Forest Classification

Abbiamo eseguito Random Forest configurando i parametri come riportato in tabella. The best estimator è rappresentato da un modello molto complesso con profondità 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

Il modello ritornato, come prevedibile, non è di supporto. Il miglioramento del test f-score, pari a 0.0036, non è significativo se paragonato ad un incremento della profondità massima di 41 livelli.

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 classifica ogni instanza in base alla classe di maggioranza dei k punti più vicini. La vicinanza è calcolata mediante la distanza euclidee. Questo rende oneroso il calcolo nel caso in cui il dataset sia composto da un moderato numero di attributi. Come possiamo leggere dalla tabella, abbiamo eseguito l'algoritmo con un massimo valore di k=12, ottenendo differenti valutazioni in funzione del numero dei vicini considerati.

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

Da una rapida analisi dei risultati osserviamo che l'f-score sui dati di training oscilla da un minimo di 0.7378 con k=1 e un massimo di 0.7684 ottenuto con k=6. Pertanto scegliamo il modello con k=6 il cui test f-score=0.9165. In generale le oscillazioni dell'f-score sui nuovi record non sono significative, variano

sensibilimente tra il 90% e il 91% (escludendo k=1), in particolare abbiamo 0.8927 con k=2 e 0.9171 con k=12. Per valori di k¿12 la f-score rimane la stessa, pertanto non abbiamo continuato ad iterare nel cercare una stima migliore.

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.

Considerando quindi paritari, i valori di f-score (escluso k=1), abbiamo cercato il valore di k avente il roc-auc (area under the curve) superiore. La tabella ci mostra che anche questa stima non subisce marcate alterazioni, ma per k=5 abbiamo una auc maggiore rispetto alle altre, pari a 0.55. Pertanto scegliamo tale k per costruire (learn) il modello con K-NN. Ci teniamo a precisare che, in questa situazione, la scelta di un k piuttosto che un altro non porta a differenze significative, in quanto tutte le stime sono molto vicine.

3.4 Miglior modello predittivo

Prima di descrivere quale tra i classificatori realizzati predice meglio sui nuovi dati è importante fare alcune precisazioni. In questo contesto è ragionevole non considerare un modello migliore rispetto ad un altro soltanto se la f-score è superiore o meno. E' importante valutare le stime nel loro insieme, quindi: accuratezza, precisione, recall e considerando l'area under the roc curve (roc-auc). Bisogna tener conto anche della complessità del classificatore, in funzione della profondità massima dell'albero con cui è stato costruito (learn). Un altro importante fattore è il costo computazionale per la realizzazione del modello: K-NN infatti per ogni record deve calcolare la distanza euclidea per punti appartenenti ad uno spazio che nel nostro caso è d 28 dimensioni.

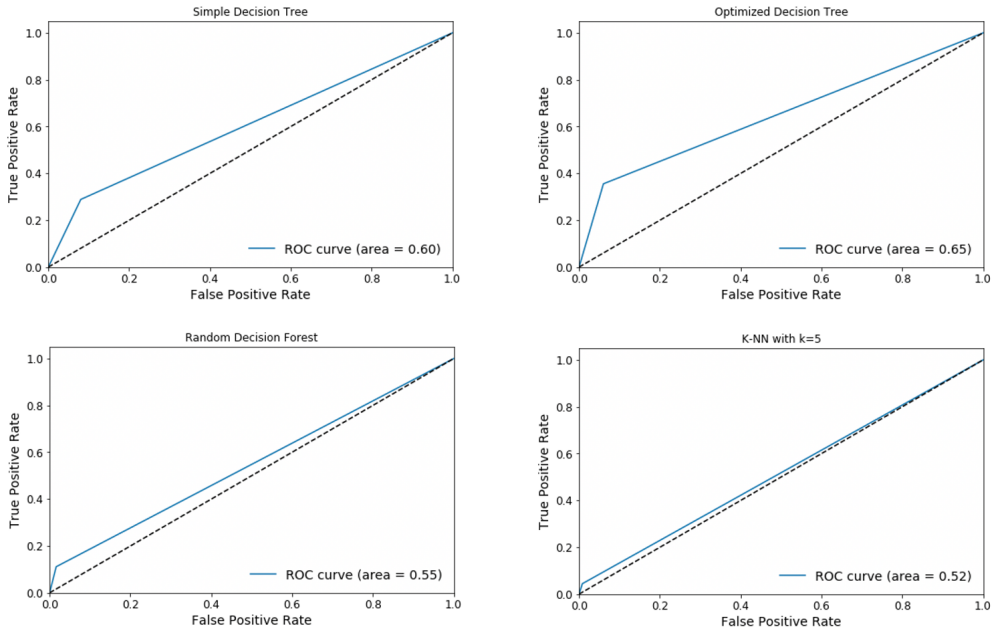


Figure 3.11: Area under the rocs curves for each model

In una prima visione di insieme possiamo affermare che, indipendentemente dal classificatore preso in esame, le stime non subiscono gravi oscillazioni (a parte recall in un caso). In particolare l'accuratezza varia meno del 2.72%, la precisione 3.01%, recall di 6.43% e f-score di 1.96%. Di conseguenza eviterei in primo luogo classificatori eccessivamente complessi come quello ottenuto con Random Forest che, nonostante abbia i valori di f-score e recall più alti (relativamente), ha una profondità massima di 46 che in questo caso non overfitta ma, su futuri nuovi dati potremmo non escluderlo: ad esempio ulteriori record relativi a nuovi dipendenti con diverse attitudini o abitudini.
Successivamente, è interessante confrontare i modelli SDT e quello realizzato con KNN con k=5. Quest'ultimo tra i vantaggi possiede la semplicità di progettazione e l'assenza di step di training, inoltre nel nostro caso

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.

ha delle buone stime e non solo per k=5. Di contro invece ha che è molto lento e il costo computazionale dovuto al calcolo della distanza euclidea in base alla dimensione del dataset (curse of dimensionality). Dall'altra parte SDT possiede invece una complessità praticamente nulla (max depth=2), le stime sono buone come in KNN ma con un discreto vantaggio nella roc-auc (fig.3.11). Queste considerazioni ci hanno portato a preferire senza esitazione SDT.
Il classificatore che secondo il nostro punto di vista ha un potenziale di predizione superiore rispetto agli altri proposti è ODT: il quale rappresenta un'evoluzione di SDT. Il modello possiede una complessità bassa in quanto l'albero ha una profondità massima di 5. Inoltre ha il valore maggiore di roc-auc pari a 0.65 e possiede i valori più alti di accuratezza e f-score, rispettivamente 85.03% e 91.40%.
In conclusione, potremmo immaginare un ordine di preferenza come segue: ODT, SDT, K-nn with k=5 and Random Forest Model.