Gianluca Massara 10730993
Carlotta Loreti 10725057
Project ID 18
Dataset 13

Data and Information Quality Project Report

Introduction

In recent years the importance of data has increased significantly. Data became important in business decision processes, in artificial intelligence but also to be analyzed to extract meaningful insights.
Due to this, also the importance of having good data quality has grown significantly. This project aims to design a data cleaning pipeline for a dataset containing information on hotels in Milan (**dataset 13**).
The final goal of the pipeline is transforming raw and unstructured data into a usable and readable format; so that it can be used for other applications.
The man steps followed have been:
- Identify the main data quality issues by means of data quality measures
- Find solutions to improve the data quality dimensions
- Recompute data quality dimensions to check whether the fixes worked or not

Data profiling and main problems

Data profiling

The dataset contains information on Milan's accommodation facilities. It contains 15 rows:
- *Ubicazione*: the location details of the hotel. It does not have a definite format but in most of the cases it contains a mixture of street, number and zd code
- *Tipo via*: the description of the type of street. It comprehends values such as *'via'* or *'viale'*
- *Descrizione via*: the actual name of the street in which the hotel is situated
- *Civico*: the address number
- *Codice via*: the code of the street
- *ZD*: the number of the city district
- *Camere*: the total number of rooms in the facility
- *Camere piano*: the field contains the number of rooms for each floor
- *Category*: the category of the hotel, in particular the number of stars
- *Insegna*: the name which is reported on the sign of the facility
- *Piani totali*: the total number of floors in the table
- *Piano piano*: contains the level at which each floor is situated
- *Posti letto*: the total number of beds in the facility
- *Posti letto per piano*: the field contains the number of beds for each floor

- *Tipo attività strutture extra*: the specific type of structure

From the columns we can already see that some columns should contain coherent values between them. For example, if we sum all values in *Camere piano* we should obtain the same value reported in *Camere*. We will explore this concept further later on in the document.
We used the ProfileReport utility of the ydata_profiling library to have an extensive and automated data profiling report: using this tool, we can see data types, value distributions and correlations between columns in a very immediate way.
We decided to use FD_MINE to check for functional dependencies in an efficient way (since it also exploits the properties of functional dependencies). We initially ran it using all attributes (excluding *Insegna* for obvious reasons) and it took forever to complete the execution (like 15 minutes); so we decided to exclude also *Civico* and *Tipo via* (that we already expect having a relationship with the other fields regarding the address of the facility) in order to reduce the execution time: now it took about 1 minute to complete and the results were basically the same we found before. By inspecting the results, we did not find any interesting functional dependency that we did not already know, so we decided to consider only the "obvious" dependencies that we could derive from the semantics of the dataset (e.g., the relationships between the fields related to the address, the relationships regarding the number of floors, the relationships regarding the number of rooms, etc.).

Main data quality issues

Given a dataset, there are different data quality dimensions which can be analyzed in order to discover if the dataset has a good level of quality.
Those dimensions are: duplicated values, completeness, accuracy, timeliness and consistency.
All those qualities were assessed in the "Data Quality Assessment" part of the notebook: consistency has been evaluated after doing some preprocessing on the dataset (i.e., summing up the ";"-separated values in *Camere piano*, *Piano piano* and *Posti letto per piano*).

Duplicated values

Given our dataset we looked for duplicate values and we found that there are no exact duplicate values in the dataset, meaning that there was no need to eliminate any duplicate row in the dataset.

Completeness

In order to obtain completeness the ratio between the number of not null cells and the total number of cells was computed.
This operation gave as a result 87.5%, which is a not so low percentage but it can be improved for sure.

Accuracy

Accuracy is defined as the extent to which data are correct, reliable and certified.

We identified that the only column in which it was possible to compute the accuracy is the *Categoria* column. We identified 6 possible values: 1, 2, 3, 4, 5, 5 STELLE LUSSO; corresponding to the possible stars.
We computed the accuracy by calculating how many values of *Categoria* fell in those values and obtaining that 438 rows out of 444 non null cells were actually accurate, resulting in a 98.6% of accuracy.
We thought that *Tipo attività strutture extra* was not suitable for an accuracy evaluation because in our opinion there may exist other types of facilities outside ‘Albergo’ and ‘Residence’ (open world assumption).

Timeliness

Timeliness is based on how much data age is appropriate for the task. In this case the dataset did not contain any timestamp, so it was not possible to compute any timeliness

Consistency

The consistency dimension captures the violation of semantic rules defined over data items. As for it we identified some semantic rules that should have been respected by the dataset which are:

- The number of elements in the array contained by *Piano Piano* field should be equal to the number reported in *Piani Totali* field; in fact, the first represents the denomination of each single floor and the second one the total number of floors. Since each floor should have a denomination they need to be equal
- The total sum of the elements contained in the *Camere Piano* array should be equal to the number reported in *Camere totali* field; in fact, the first one indicates the number of rooms for each floor and the second one the total number of rooms of the accommodation facility
- The same as above but with *Posti letto per piano* and *Posti letto* since the former represents the number of beds for each floor and the latter represents the total number of beds in the facility

In both cases we used plots to better visualize those data and the number of discrepancies is very high: for the first rule there is a consistency of 86.3%, while for the second rule and the third rule there is a consistency of, respectively, 23.9% and 20.2%, which is a really low result.

Data Quality Pipeline

Data Quality Assessment

The Data Quality Assessment to identify which are the main data quality issues of the data set has been the first operation performed.
In order to compute completeness, accuracy and consistency the standard formulas were used; in particular:

- To compute completeness we applied $\frac{non\ empty\ cells}{total\ number\ of\ cells}$
- To compute accuracy we applied $\frac{correct\ cells}{not\ null\ cells\ of\ the\ column}$

- To compute consistency we applied $\frac{cells\ satisfying\ the\ property}{total\ cells\ of\ the\ column}$

The results have already been described in the previous section.

Data Standardization

The given dataset did not contain any column with a measurement unit, so there was no need to perform data unit standardizations (for example going from entries having all different currencies, to having all entries expressed in a single currency.
However, there were other standardizations that needed to be performed.
We first started by looking at the Piano *piano* column and we observed that there were some entries that had “T” and some that had “0”, both representing the ground floor; so we decided to keep only one of those two representations and we chose “0”. So, all “T”s were replaced by zeros.
In the same column we noticed also that there were some entries containing only semicolumns, so we decided to interleave the columns with incrementing numbers. For example “;;;” became “0;1;2;3”.
Then we moved on to the *Categoria* column. The column should have values ranging from “1” to “5L” meaning 5 Stars Luxury, which is the highest category. In the dataset we found some other values which were “l” and “5 Stelle Lusso”; values have been replaced by “5L”.
Next, the *Tipo attività struture extra* column has been replaced by *Tipo attività strutture extra* which is more correct and the cells containing “albergo” as a value were corrected with the value “Albergo” so that the formatting of the word is the same for the whole column.

Handling of Missing Values

The first action performed in this stage is the drop of useless rows. We considered rows useless if the percentage of null values is greater or equal to 80%. Performing an analysis on the dataset we discovered that only one row could be considered useless, which is row 322 having only the *Ubicazione* value not null. Even if from the *Ubicazione* column it is possible to extract other information on the facility, we decided to drop it anyway; in fact, if we extracted data from *Ubicazione* the null percentage would go down only to 66% which is still pretty high.
Next we focused on the *Categoria*, *Tipo attività strutture extra* and *Insegna* columns where we replaced null values with “Not Specified”. This is because we considered those type of columns as not suitable for imputation, so the best solution we found to not leave null values was to set them to not specified.
After that, we went on fixing the null values in *Tipo via*, *Descrizione via*, *ZD* and *civico* columns. We noticed that null values in those columns could be filled with data extracted from the *Ubicazione* column.
As for *Tipo via* column we first started by making a list of the possible values that the column can have which resulted in being ['ALZ', 'CSO', 'GLL', 'LGO', 'PLE', 'PZA', 'VIA', 'VLE']. Then we made a list of all those rows that have the first part of the *Ubicazione* column, a value contained in the previously mentioned list and a null value in the *Tipo via* column. For those rows we filled the null value with the first part of *Ubicazione*.
To extract data for *Descrizione via*, *ZD* and *Civico* columns we used regular expressions and filled the corresponding null values with the newly found values. We chose to adopt this method

because the rows in *Ubicazione* are formatted many different ways and it would have been more difficult to perform this action by splitting the string.
We then dropped 3 rows that still had null values in either of the fields we dealt with just above: we did so because there were no other rows that could help us in imputing those values and because we felt that missing a part of the address would make the facility not recognizable (and thus unusable).
After performing all those extractions, we noticed that there are some rows that share the same value of *Descrizione via* but some of those rows have “nan” *Codice Via*. In order to solve this problem, we grouped the rows by their *Descrizione via* value and we filled nulls with the most frequent value. Then we checked which rows still had the *Codice via* null; those rows could not be imputed as we did not have any reference on how to fill them, so we set their *Codice via* to 0.
In the process all values that could be casted as int were casted in order to have the right format of data.

Error Correction (pt.1)

In this section of the pipeline we dedicated to correct errors and inconsistencies that were present in the dataset.
First thing we did was notice that there are some rows that share the same *Descrizione via* but have different *Tipo via* values. To solve the problem we grouped rows based on *Descrizione via* field and the we computed, for each group, which was the most frequent value of *Tipo via*, this value will be the one chosen to be true

Data Transformation

In this section of the pipeline we focused on improvements that could be made in the dataset.
First, we started by reformatting the *Ubicazione* column using the values of the columns *Tipo via, Descrizione via, Civico* and, where present, *ZD*. In this way all rows of *Ubicazione* look the same.
Next we casted *Camere* and *Posti Letto* to int as it’s the right type of data needed for those columns.
Last we set *Insegna* and *Tipo attività strutture extra* all in upper case as most of them were already like this.

Missing Values Imputation and Error Correction (pt.2)

In this section of the pipeline we imputed missing data and then we corrected inconsistencies using the results of the imputation.
We decided to impute with a ML technique only *Piani totali* columns and then extract other missing information using data already available.
To impute *Piani totali* we used the MICE technique because it is a robust method that, by running multiple iterations, imputes missing values maintaining the statistical properties of the dataset and not introducing any bias.
After imputing *Piani totali*, we checked if the imputed values were coherent with *Piano Piano* data, if available. If they were coherent they were marked as valid and then input in the original dataset. For those rows which still have missing values, so those which were marked as not valid, we used the data of *Piano piano* to fill missing values in *Piani totali.*

We checked how many rows were left with mismatching values of *Piano piano* and *Piani totali*: 12 rows are not coherent. Out of those 12 rows we observed that 11 of them have coherent values of *Piano piano, Camere piano* and *Posti letto per piano* (meaning that the values for those columns in a row have all the same length), so we updated the value of *Piani totali* using the length of those fields.
Now only one row still has conflicting values and, since there is no way to know which is the true value, we took the *Camere piano* length as the correct value for *Piani totali*.
We then checked the coherence between *Camere piano* and *Piani totali* (which was quite low, 35.8%) and we did the following reasonings:

- If the sum of the values in *Camere piano* is equal to the value of *Camere*, then we assume that the number of floors that we can extract from *Camere piano* is correct
- If the value of *Camere* is equal to the product of the values of *Camere piano* and *Piani totali*, then we can rewrite *Camere piano* according to our standard; the same also holds for *Posti letto* and *Posti letto per piano*: in this case we also regard as true the value of *Camere* and we recompute *Camere piano*
- Null values in *Camere piano* can be filled using the information contained in *Camere* and *Piani totali* by distributing the rooms on the floors

Now, the remaining rows with conflicting values are rows where *Camere piano* is not null but it is also not coherent with *Camere* and *Piani totali*: we decided to consider the latter columns as true and to recompute *Camere piano* accordingly (by adding the missing floors and redistributing the missing rooms). However, we also had to update *Posti letto* and *Posti letto per piano*: in fact, now there were floors that have more rooms than beds, so we added beds to these floors.
We also dealt with the inconsistency regarding *Posti letto* and *Posti letto per piano*: in case the two were not coherent, we considered as true the value of *Posti letto* and we recomputed the other field (by adding beds when beds were missing, and by giving to each floor the average number of beds per floor when there were more beds than expected or in case of null values).
We then detected a row that already had inconsistencies regarding the number of floors in the original data: *Posti letto per piano* had one more floor, so we decided to merge the number of beds of the last floor.
Then we corrected inconsistencies regarding the rooms:

- If the sum of the values in *Camere piano* is smaller than the value of *Camere*, we added the missing rooms to the last floor. We then had to adjust again *Posti letto* and *Posti letto per piano*, because there was the risk of having again more rooms than beds per floor
- Otherwise, we trusted the values in *Camere piano* and we updated *Camere*: again, we needed to adjust *Posti letto* and *Posti letto per piano*

Finally, we checked whether for each floor the number of rooms is smaller than the number of beds: the consistency was very high (97.1%), but we still had some rows that were still inconsistent. In fact, we had two cases:

- On some floors there were no rooms but there were beds: we redistributed the beds to the other floors
- Some rows still had more rooms than beds: we decided to update *Posti letto* and *Posti letto per piano* so that the constraint is satisfied

Outlier Detection

We used histograms to visualize the distribution of *Piani totali*: we chose histograms because the values belonged to a limited range that was easily visualizable. The most frequent value is 4, as we expected, because it is the result of the imputation process; in our opinion, there are not really any outliers in this sense since facilities with a high number of floors are few (as in the real world).
Then, we used scatter plots to visualize the relationships between *Camere* and *Piani totali*, *Camere* and *Posti letto*, and *Posti letto* and *Piani totali*. We saw the following results:

- There were some facilities with just one floor but more than 50 rooms. We checked and these outliers were due to the fact that originally in these rows *Camere piano* simply contained the same value as *Camere* (and the same goes for *Posti letto per piano* and *Posti letto*), while missing the information about *Piani totali* and *Piano piano*: according to what we have done above, the value of *Piani totali* is assumed to be 1. However, there was little we could do differently since there was no other information we could extract about floors: the only other option was using the result of the imputation process, but we decided to trust the already present information.
- There were facilities where some floors had strange beds/room ratios (i.e., > 4): this was due to the fact that, while dealing with inconsistencies, we redistributed beds without considering this aspect.

In both cases, we decided not to drop outliers.

Data deduplication

We used the Jaro-Winkler similarity measure to compare rows based on their *Insegna* values since that should be a unique column combination. To reduce the number of comparisons to be performed, we used *Descrizione via* as a blocking key; we also excluded common words, such as “hotel” and “albergo”, from the comparison since otherwise we would have had a lot of false positives.
We found 7 pairs of possible duplicates and we decided that

- If the values of *Civico* or the values of *Tipo attività strutture extra* are different, the tuples do not match: in fact, we thought that maybe a facility could occupy two different civic numbers in the same street or the same property (hence the same name) could have a hotel and a residence in the same street (or even in the same building)
- Otherwise, the tuples match and we decided to keep the row with the highest index (since it may represent a “fresher” and more accurate value) and to drop the other

Final Data Quality Assessment

We recomputed the dimensions we computed at the beginning: both completeness and consistency now yield perfect results (100%).
Accuracy regarding *Categoria* has improved (now it is 98.7%) but it does not have a perfect score because of some ‘Not specified’ values.