

**Data Cleaning in Health Sector**

Student: Nikolaos Moutsopoulos

Ε-10102

Contents

[1. Introduction 3](#_Toc69423295)

[2. Data Mining 5](#_Toc69423296)

[2.1 Introduction 5](#_Toc69423297)

[2.2 Steps of Data Mining 6](#_Toc69423298)

[2.3 Data Mining techniques 8](#_Toc69423299)

[3. Data Cleaning 9](#_Toc69423300)

[3.1 Introduction 9](#_Toc69423301)

[3.2 Data Cleaning Approcahes 11](#_Toc69423302)

[3.3 Data Cleaning in Health Care 14](#_Toc69423303)

[4. Our Approach 16](#_Toc69423304)

[4.1 Introduction 17](#_Toc69423305)

[4.2 Dataset Presentation 17](#_Toc69423306)

[4.2.1 United States Big Cities Data 17](#_Toc69423307)

[4.2.2 United States Hospitals - Cancer Events 19](#_Toc69423308)

[4.3 Overall Pipeline 20](#_Toc69423309)

[4.4 Load Data 22](#_Toc69423310)

[4.5 Dataset Format 23](#_Toc69423311)

[4.6 Remove Duplicates/Nulls 24](#_Toc69423312)

[4.7 Fill Null Values 25](#_Toc69423313)

[4.8 Outlier Detection 26](#_Toc69423314)

[4.8.1 Statistical Method 26](#_Toc69423315)

[4.8.2 Algorithmic Method 27](#_Toc69423316)

[4.9 Plots 28](#_Toc69423317)

[4.10 Transform Data for NLP Processing 30](#_Toc69423318)

[4.11 Evaluation/Outcomes 31](#_Toc69423319)

[5. Conclusion 32](#_Toc69423320)

# 1. Introduction

Data collection is of paramount importance for large organizations not only for record keeping. Organizations need to support a variety of data analysis tasks that are critical to their mission, where data analysis typically drives decision-making processes and efficiency optimizations.

However, despite the importance of data collection and analysis, data quality remains a pervasive and on-going problem in almost every large organization because the presence of incorrect or inconsistent data can significantly distort the results of analyses, often negating the potential benefits of information-driven approaches (Hellerstein, 2008).

The quality of quantitative data can be looked from (a) a statistical perspective with emphasis on outlier detection or (b) checked through the use of algorithms and implementations that can be implemented in large databases (Hellerstein, 2008).

Nowadays more and more industries and organizations use data for various reasons such as take important decisions or sell algorithms as products. For example, the healthcare industry which is turning into a developed and complex marketplace uses data collection tools and methods which appear to be crucial sources for generating information about patients. This information is supposed to improve the overall quality healthcare approach, and deliver care according to patient needs. However, poor data quality in healthcare is the number one issue that requires major improvement especially if it is considered that decisions in this industry could literally mean the difference between life and death. This data, especially when related to healthcare, cannot be wrong, inaccurate, incomplete or unrecognizable to the operations and processes that consume them. The ramifications of inaccurate data could impact patient safety, accurate reimbursement for services, and many other aspects of healthcare delivery. Other fields where data collection plays a major role include environmental, industrial, surveillance, computer network, biological, astronomy, web, information network and economics applications (Gupta *et al.,* 2014).

Databases also play an important role in today’s IT-based economy. Many industries and systems depend on the accuracy of databases to carry out operations. Therefore, the quality of the information (or the lack thereof) stored in the databases can have significant cost implications to a system that relies on information to function and conduct business. In an error-free system with perfectly clean data, the construction of a comprehensive view of the data consists of linking—in relational terms, joining—two or more tables on their key fields. Unfortunately, data often lack a unique, global identifier that would permit such an operation. Furthermore, the data are neither carefully controlled for quality, nor defined in a consistent way across different data sources (Elmagarmid *et al.,* 2007). Thus, data quality is often compromised by many factors, including data entry errors (e.g., Microsft instead of Microsoft), missing integrity constraints (e.g., allowing entries such as Employee Age 1⁄4 567), and multiple conventions for recording information (e.g., 44 W. 4th St. versus 44 West Fourth Street). Often, while integrating data from different sources to implement a data warehouse, organizations become aware of potential systematic differences or conflicts. Such problems fall under the umbrella-term *data heterogeneity*, *Data* *cleaning*, or *data scrubbing*, the latter ones referring to the process of resolving such identification problems in the data.

Taking the above into consideration, this paper focuses mostly on data cleaning methods. It is clear that data cleaning (data cleansing) methods should be applied in order to control the data integrity and quality (Pipino *et al.,* 2002). Consequently, there has been a variety of research over the last decades on various aspects of anomaly detection and data errors and subsequent data cleaning.

# 2. Data Mining

## 2.1 Introduction

Data cleaning is considered as a part of a larger process of data gathering and exploitation which is called data mining.

Data mining is the process of gathering and cleaning raw data, finding patterns, creating models, and testing those models in order to turn raw data into useful and meaningful information for businesses and organisations. These actions require knowledge in various fields such as statistics and machine learning.

Organizations use data mining to cover some of their needs such as database marketing, credit risk management, [fraud detection](https://www.investopedia.com/financial-edge/0512/the-most-common-types-of-consumer-fraud.aspx), spam Email filtering, or even to discern the sentiment or opinion of users.

Data mining is based on three basic concepts to produce results, association rules, support and confidence. [Association rules](https://searchbusinessanalytics.techtarget.com/definition/association-rules-in-data-mining) are created by analyzing data for frequent if/then patterns, then using the support and confidence criteria to locate the most important relationships within the data. Support is how frequently the items appear in the [database](https://searchsqlserver.techtarget.com/definition/database), while confidence is the number of times if/then statements are accurate.

Other data mining [parameters](https://whatis.techtarget.com/definition/parameter) include Sequence or Path Analysis, [Classification](https://searchdatamanagement.techtarget.com/definition/data-classification), [Clustering](https://whatis.techtarget.com/definition/cluster) and Forecasting. Sequence or Path Analysis parameters look for patterns where one event leads to another later event. A Sequence is an ordered list of sets of items, and it is a common type of data structure found in many databases. A Classification parameter looks for new patterns, and might result in a change in the way the data is organized. Classification [algorithms](https://whatis.techtarget.com/definition/algorithm) predict variables based on other factors within the database.

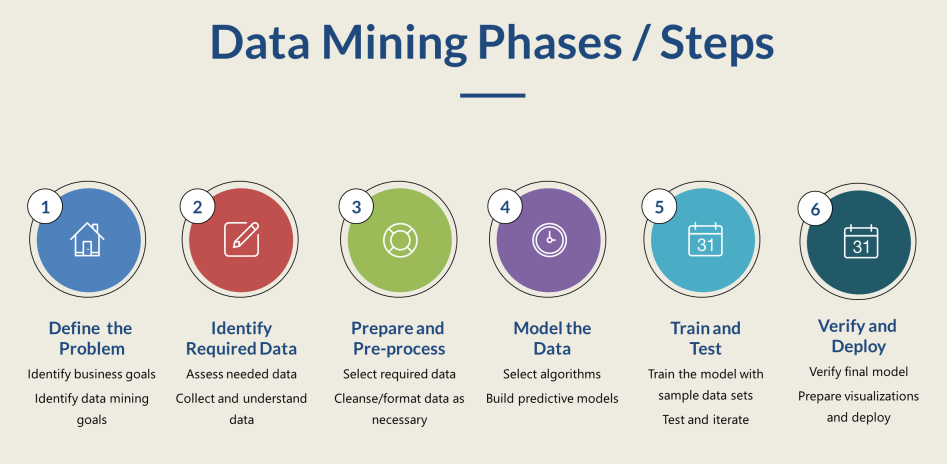
Data mining events are present in our every day life. A well known example is when a person visits an e-shop and creates an account, there is a tracking algorithm which records the actions of the person during the visit. Those data are stored in a Data Warehouse. Later the marketing department can take those data into consideration and send an offer to their potential or current customer on the items that he visited more frequently.

## 2.2 Steps of Data Mining

The first step in data mining is almost always data collection. Today’s organizations can collect records, logs, website visitors’ data, application data, sales data, and more every day. Collecting and mapping data is a good first step in understanding the limits of what can be done with and asked of the data in question.

The Cross-Industry Standard Process for Data Mining (CRISP-DM) is an excellent guideline for starting the data mining process. This standard was created decades ago and is still a popular paradigm for organizations that are just starting.

The 6 CRISP-DM phases



The CRISP-DM comprises a six-phase workflow. It was designed to be flexible; data teams are allowed and encouraged to move back to a previous stage if needed. The model also provides opportunities for software platforms that help perform or augment some of these tasks.

**1. Business understanding**

Comprehensive data mining projects start by first identifying project objectives and scope. The business stakeholders will ask a question or state a problem that data mining can answer or solve.

**2. Data understanding**

Once the business problem is understood, it is time to collect the data relevant to the question and get a feel for the data set. This data often comes from multiple sources, including structured data and unstructured data. This stage may include some exploratory analysis to uncover some preliminary patterns. At the end of this phase, the data mining team has selected the subset of data for analysis and modeling.

**3. Data preparation**

This phase begins with more intensive work. Data preparation involves preparing the final data set, which includes all the relevant data needed to answer the business question. Stakeholders will identify the dimensions and variables to explore and prepare the final data set for model creation.

**4. Modeling**

In this phase, you’ll select the appropriate modeling techniques for the given data. These techniques can include clustering, predictive models, classification, estimation, or a combination. [Front Health used statistical modeling and predictive analytics](https://www.tableau.com/solutions/customer/front-health-improves-clinical-quality-patient-health-cost-of-care-with-tableau) to decide whether to expand healthcare programs to other populations. You may have to return to the data preparation phase if you select a modeling technique that requires selecting other variables or preparing some different sources.

**5. Evaluation**

After creating the models, you need to test them and measure their success at answering the question identified in the first phase. The model may answer facets of things not accounted for, and you may need to edit the model or edit the question. This phase is designed to allow you to look at the progress so far and ensure it’s on the right track for meeting the business goals. If it’s not, there might be a need to move backwards to previous steps before a project is ready for the deployment phase.

**6. Deployment**

Finally, once the model is accurate and reliable, it is time to deploy it in the real world. The deployment can take place within the organization, be shared with customers, or be used to generate a report for stakeholders to prove its reliability. The work doesn’t end when the last line of code is complete; deployment requires careful thought, a roll-out plan, and a way to make sure the right people are appropriately informed. The data mining team is responsible for the audience’s understanding of the project.

## 2.3 Data Mining techniques

Data mining includes multiple techniques for answering the business question or helping solve a problem. This section is just an introduction to [two data mining techniques and is not currently comprehensive.](https://www.tableau.com/learn/articles/introduction-data-mining-techniques)

**Classification**

The most common technique is classification. To do this, identify a target variable and then divide that variable into appropriate level of detail categories. For example, the variable ‘occupation level’ might be split into ‘entry-level’, ‘associate’, and ‘senior’. With other fields such as age and education level, you can train your data model to predict what occupation level a person is more likely to have. You may add an entry for a recent 22-year-old graduate, and the data model could automatically classify that person in an ‘entry-level’ position. Insurance or financial institutions such as PEMCO Insurance used classification to [train their algorithms to flag fraud and to monitor claims.](https://www.tableau.com/solutions/customer/PEMCO-insurance-claims-handling-server-management-with-tableau)

**Clustering**

Clustering is another common technique, grouping records, observations, or cases by similarity. There won’t be a target variable like in classification. Instead, clustering just means separating the data set into subgroups. This method can include grouping records of users by geographic area or age group. Typically, clustering the data into subgroups is preparation for analysis. The subgroups become inputs for a different technique.

# 3. Data Cleaning

## 3.1 Introduction

Data cleaning is the identification and correction of corrupted, duplicate, missing or inaccurate data. The capability of the organization to operate efficiently and to make accurate decisions that lead to positive outcomes requires these activities and processes be engrained in daily operations. Data cleaning has played a critical role in ensuring data quality for enterprise applications. Naturally, there has been extensive research in this area, and many data cleaning algorithms have been translated into tools to detect and to possibly repair certain classes of errors such as outliers, duplicates, missing values, and violations of integrity constraints (Abedjan *et al.* 2016).

In many if not most instances, data can only be cleaned effectively with some human involvement. Therefore, there is typically an interaction between data cleaning tools and data visualization systems (Hellerstein, 2008). Almost all practical tools involve humans, for example, to verify detected errors, to specify cleaning rules, or to provide feedback that can be part of a machine learning algorithm (Abedjan *et al.* 2016).

Data cleaning involves the use of computational procedures to automatically or semi automatically identify and, when possible, correct errors in large data sets (Hellerstein, 2008). Data cleaning methods can focus on errors in quantitative attributes of large databases, as well as on other types of attributes.

Most of the research has focused on approaches that examine the side effects as a proxy to errors, also research has been done on outlier detection processes which investigate data points that deviate significantly with respect to their underlying distribution.

However, most existing discovery techniques have traditionally ignored the time dimension. Recurrent events, such as persons reported in locations, have a duration in which they are valid, and this duration should be part of the rules or the cleaning process would simply fail.

Therefore, it is very important to also look at the rule discovery problem for temporal web data. Such a discovery process is challenging because of the nature of web data; extracted facts are (i) sparse over time, (ii) reported with delays, and (iii) often reported with errors over the values because of inaccurate sources or non-robust extractors (Abedjan *et al.* 2015).

In general, the current state of available data cleaning solutions and tools belong to one or more of the following four categories according to (Abedjan *et al.* 2016):

* *Rule-based detection algorithms* that can be embedded into frameworks, such as NADEEF, where a rule can vary from a simple “not null” constraint to multi-attribute functional dependencies (FDs) to user-defined functions. By using this class of tools, a user can specify a collection of rules that clean data will obey, and the tool will find any violations.
* *Pattern enforcement and transformation tools* such as OpenRefine, Data Wrangler and its commercial descendant TRIFACTA, KATARA, and DataX- Former. These tools discover patterns in the data, either syntactic (e.g., OpenRefine and Trifacta) or semantic (e.g., Katara), and use these to detect errors (cells that do not conform with the patterns). Trans- formation tools can also be used to change data representation and expose additional patterns.
* *Quantitative error detection algorithms* that expose outliers, and glitches in the data.
* *Record linkage and de-duplication algorithms* for detecting duplicate data records, such as the Data Tamer system and its commercial descendant TAMR. These tools perform entity consolidation when multiple records have data for the same entity. As a side effect of this process, conflicting values for the same attribute can be found, indicating possible errors.

## 3.2 Data Cleaning Approcahes

The simplest case to consider, and one of the most useful, is to analyze the set of values that appear in a single column of a database table. Many sources of dirty quantitative data are discoverable by examining one column at a time, including common cases of mistyping and the use of extreme default values to achieve spurious integrity on numeric columns. In general, quantitative data cleaning relies on the use of statistical methods to identify and repair data quality problems (Prokoshyna et al., 2015)

There are various ways to make this notion concrete, which rest on defining specific metrics for the center of the set of values (what is average) and the dispersion of the set (which determines what is far from average, in a relative sense). The center, or core, of a set of values is some typical value that may or may not appear in the set. The most familiar center metric is the mean (average) of the values, which typically is not one of the values in the set (Hellerstein, 2008).

This single-attribute case provides an opportunity to introduce basic statistical concepts in a relatively intuitive setting. A notion of outliers based on some intuitive statistical properties can be subsequently developed and then analogs to those properties can be described that can remain robust, even when significant errors are injected into a large fraction of the data (Hellerstein, 2008).

The dispersion, or spread, of values around the center gives a sense of what kinds of deviation from the center are common. The most familiar metric of dispersion is the standard deviation, or the variance, which is equal to the standard deviation squared. So, for example, a typical definition of an outlier is any value that is more than 2 standard deviations from the mean.

The center/dispersion about outliers defines one of the most familiar ideas in statistics: the normal distribution, sometimes called a Gaussian distribution, and familiarly known as the bell curve. Normal distributions are at the heart of many statistical techniques, especially those that focus on measuring the variation of errors. The normal distribution is defined by a mean value and a standard deviation, and has the probability density function:



A third class of metrics that is often discussed, is the skew of the values, which describes how symmetrically the data is dispersed around the center. In very skewed data, one side of the center has a much longer “tail” than the other (Hellerstein, 2008).

A statistical subfield is referred to as robust statistics that considers the effect of corrupted data values on distributions, and develops estimators that are robust to such corruptions. Robust estimators can capture important properties of data distributions in a way that is stable in the face of many corruptions of the data, even when these corruptions result in arbitrarily bad values. When the percentage of corruptions in a data set exceeds a threshold called the breakdown point of an estimator, the estimator can produce arbitrarily erroneous results. Robust metrics, are also needed for the dispersion or spread of the distribution. A good robust metric of dispersion is the Median Absolute Deviation (MAD), which is a robust analogy to the standard deviation, measuring the median distance of all the values from the median value. (ref 10). The median and MAD lead to a robust outlier detection technique known as Hampel X84 which is considerd quite reliable because it can be shown to have an ideal breakdown point of 50 %.

In addition to the above, for very large tables and and/or scenarios where efficiency is critical, there are algorithms in the database research literature that can, in a single pass of a massive data set, compute approximate values for the median or any other quantile, using limited memory. The approximation is in terms of the rank order: rather than returning the desired value (median, quantile, etc.), it will return some value from the data set that is within εN ranks of the desired value. The two most-cited algorithms (Manku et al., 1998, Greenwald and Khanna, 2001) differ slightly in their implementation and guarantees, but share the same general approach. Both work by scanning the column of data and storing copies of the values in memory along with a weight per value; during the scan, certain rules are used to discard some of the values in memory and update the weights of others. At the end of the scan, the surviving values and weights are used to produce an estimate of the median (or quantiles).

Again, these algorithms can be implemented by code running outside the database engine to manage the discard and weighting process. Unfortunately, that approach will transfer every value from the database table over to the program running the algorithm which can be very inefficient, particularly if the median-finding program is running across a network from the server. To avoid this, and provide better software modularity, some modern databases allow user-defined aggregate (UDA) functions to be registered with the database. Once a UDA is registered, it can be invoked conveniently from SQL and executed within the server during query processing.

In cases where data sets are not normally distributed, there is a high probability that the outlier detection schemes mentioned above will work reasonably. These distributions are referred to as Multimodial distributions where a data set can have many peaks or Zipfian distributions where a large fraction of the data is condensed into a small fraction of values, with the remainder of the data spread across the “long tail” of rare values (Hellerstein, 2008).

In some cases, in order to get reliable outlier detection, resampling methods might be needed. Examples of resampling methods include the basic bootstrap method and the jackknife method. The basic idea behind resampling, is to repeatedly take samples of a data set in a controlled fashion, compute a summary statistic over each sample, and carefully combine the samples to estimate of a property of the entire data set. Intuitively, rare outliers will appear in few or no samples, and hence will not perturb the estimators used. Given good estimators computed in this fashion, outliers can be found by their divergence from the estimated properties of the data.

Other outlier methods focus on two extremes of frequencies: distinct attributes where nearly every value has frequency 1, and attributes that have high frequency spikes at some value. It is often the case in dirty data that a would-be key contains some duplicated values, due either to data entry errors, or repeated use of some value as a code for “unknown". The simple scenario for data repair, will be to have knowledge about the column(s) that form a key, and try to determine which entries in the key column(s) need to be cleaned. The more complex scenario is “key discovery”, where one is trying to discover which columns might actually be intended as keys, despite the fact that there may be some dirty data. This problem requires coming up with a metric for how “close” a column is to being a key. One intuitive measure for dirty keys is what we call the unique row ratio: the ratio of distinct values in the column to the total number of rows in the table. If this is close to 1.0, the column may be flagged as a potential dirty key (Dasu et al., 2002). Another measure for dirty keys is an efficient algorithm called Tame for discovering approximate composite keys; it can be used with any of the metrics we describe above. The core of Tane is an efficient algorithm for deciding which combination of columns to text for \key-ness" next, based on the combinations previously tested.

Apart from the methods mentioned, there are some classic techniques that one can use in data cleaning. The human eye can be used as the analysis engine to visualize data. Histograms are a natural way to visualize the density of data points across values of a single dimension. The basic idea of a histogram is to partition the data set into bins, and plot the count or probability of items in each bin. Extreme outliers can often be seen easily in equi-width histograms as bins at the far right and left, as can extreme frequency outliers (very tall bins). However, extreme outliers can also swamp more detailed display of other data. Equi-depth histograms construct bins of near-equal count (depth), but the bins differ in the width of their endpoints (Greenwald et al., 2001).

## 3.3 Data Cleaning in Health Care

Healthcare is a domain in which data have a great value and their usage may give very benficial result to humanity. More and more the healthcare industry embraces techniques regarding data gathering from various sources which later are being analyzed and producing results in order to find better curing methods or predict the health situation of a person regarding his habits.

Healthcare systems collect, analyze, and share protected healthcare information (PHI) every day, but it’s not always accurate or properly structured. To ensure the portability, accessibility, and interoperability of such information, healthcare data cleaning is often a necessity.

Typically, most organizations store data in databases. These could be associated with your Electronic Health Record (EHR), decision support system, revenue cycle management, and many more applications designed to enable the healthcare ecosystem to work more cohesively. The value of [healthcare big data](https://infowerks.com/healthcare-big-data-strategy/) is immense, helping improve care, boost revenue, and drive better decision-making. Dirty data makes that virtually impossible.

Dirty data describes information that is inaccurate, outdated, redundant, incomplete, or formatted incorrectly. Using healthcare data cleaning, you can bring consistency to your data. This consistency is necessary when integrating disparate streams of data. If you merge dirty data, then its ability to be actionable is lost.

In an ideal world, all [healthcare information systems](https://infowerks.com/health-information-system/) (HIS) would work together in harmony. Field matching wouldn’t be a roadblock, nor would duplicates or other inconsistencies. Unfortunately, that’s just not the case. There is currently no standardized practice for healthcare data interoperability.

.However, it’s still not as easy as moving data from one system to another or quickly aggregating different data sets and automatically have a working process. As healthcare data management experts, we see on a daily basis how difficult it is to map data from one system to another, even when they are in the same category. So, if you can adeptly move from one EHR to another, then it gets really tricky when combining data outputs or moving information into a completely different type of platform.

Dirty data is not the result of one thing; it’s a culmination of lots of factors, some more significant than others. One of the biggest concerns is duplication. According to research, duplicate records make up [5-10% of a hospital’s EHR](http://bok.ahima.org/doc?oid=302567#.XrMHVqhKiUk). That number expands to rates of 20% for healthcare entities that have multiple locations.

Duplications happen for many reasons, including errors in spelling or other patient data. Depending on the parameters of the system, it may be unable to search for duplicates as new patients are added.

Another symptom of dirty data is that it’s incomplete. Without all the appropriate fields, records may be useless. If a patient record list omits things like preexisting conditions or allergies, it’s not only incomplete but could impact care. Incomplete information can be attributed to user error or system limitations.

The third significant cause of dirty data is inaccuracies. Errors might have occurred in the original set-up (i.e., misspelled names, transposed numbers), or the data may not have been updated correctly. If you don’t have accurate information about your patients, from contact information to insurance codes, then it’s harder to communicate with them and leverage your information for better outcomes and insights.

The consequences of dirty data can be numerous. First, there are the monetary losses. Gartner researchers revealed that the cost of poor data equates to [$9.7 to $14.2 million](https://www.gartner.com/smarterwithgartner/how-to-stop-data-quality-undermining-your-business/) for businesses every year. Those numbers reflect all types of companies, but it’s still an important figure to know.

Where do these losses come from? For healthcare, it could be from several things, such as opportunity costs associated with being able to launch new applications to the higher hard costs of unpaid reimbursements from payers and additional labor needed to strip out the bad data.

The costs are more than fiscal. You’ll lose time because you can’t seamlessly convert data into new platforms. You’ll miss out on insights that could help you find ways to cut costs and work more efficiently. Worst of all, it could impact patient care.

# 4. Our Approach

## 4.1 Introduction

As we already have mentioned the health care is an industry where data mining has a great value and mainly its scope intends to offer us a better standard of living. Consequently we focus our research on this industry in order to gain knowledge about how different types of data cleaning techniques affect the cleaning process.

The thesis’s scope, more specifically, focuses on two different techniques which are used to clean two different datasets. Namely during the project we used statistical and algorithmic data cleaning methods.

The programming language used for the project is python and its libraries regarding data cleaning and data analysis.

Below will be described in detail the datasets that we worked on and the work that we made on them in order to load it on python dataframes and proceed with data cleaning.

Also will be described the methods that we used in each step of the data cleaning process, will be displayed some plots that we used to understand data anomalies or outcomes.

Finally will be described our results about the effectiveness and the quality of the outcome of each method used.

## 4.2 Dataset Presentation

As we have already mentioned for the project scope we used two real life datasets produced by the health care industry. Below will be examined separately each one of them in order to explain their content, the problems that we cope with and the outcome that we want to produce.

### 4.2.1 United States Big Cities Data

The dataset that will be examined below is a .csv file and illustrates health status of 26 of the nation’s largest and most urban cities as captured by 34 health (and six demographics-related) indicators.

These indicators represent some of the leading causes of morbidity and mortality in the United States and leading priorities of national, state, and local health agencies. Public health data were captured in nine overarching categories: HIV/AIDS, cancer, nutrition/physical activity/obesity, food safety, infectious disease, maternal and child health, tobacco, injury/violence, and behavioral health/substance abuse.

Below displayed a screenshot from the header and the first 5 records of the dataset in order to give a better view of the data that we have in our disposal.



This dataset is consisted of 9 columns:

* Indicator Category
* Indicator
* Year
* Gender
* Race/ Ethnicity
* Value
* Place
* BCHC Requested Methodology
* Source
* Methods
* Notes

One of the first decisions we made was to remove the last four columns from our study, namely: *BCHC Requested Methodology, Source, Methods, Notes.* Although those four columns contain useful info, in raw text, regarding the source or some extra info about our indicators we decided that there was not value to cut or aggregate any data for these. Despite the fact that we did not make any data cleaning process, we managed to apply methods that would prepare those columns to be able to read by an NLP algorithm.

On the other hand we focused on cleaning our valuable data. We emphasized in removing duplicate values, filling some null values in order to receive a more accurate result and we narrowed down the data in order to produce our reports with the indicators that had more value for us.

### 4.2.2 United States Hospitals - Cancer Events

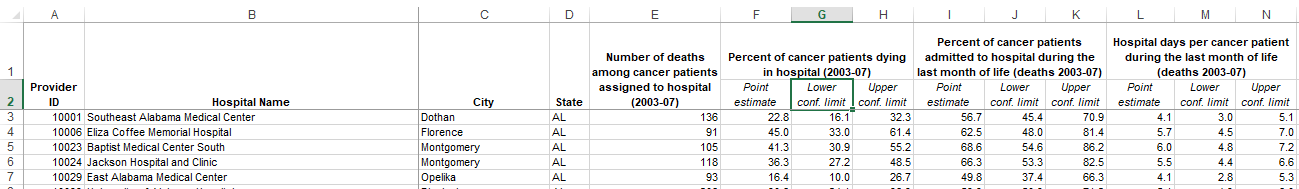
The second dataset is an .xls file which illustrates specific events about cancer patients from 938 hospitals of the United States.

This dataset gives information about the hospital name along with the city, the state, the number of deaths among cancer patients and 11 eleven other indicators which are displayed with their point estimate, the lower conf. limit and the upper conf. limit. Below the indicators will be described distinctively.

* Percent of cancer patients dying in hospital (2003-07)
* Percent of cancer patients admitted to hospital during the last month of life (deaths 2003-07)
* Hospital days per cancer patient during the last month of life (deaths 2003-07)
* Percent of cancer patients admitted to intensive care during the last month of life (deaths 2003-07)
* ICU days per cancer patient during the last month of life (deaths 2003-07)
* Percent of cancer patients receiving life-sustaining treatment during the last month of life (deaths 2003-07)
* Percent of cancer patients receiving chemotherapy during the last two weeks of life (deaths 2003-07)
* Percent of cancer patients enrolled in hospice during the last month of life (deaths 2003-07)
* Hospice days per cancer patient during the last month of life (deaths 2003-07)
* Percent of cancer patients enrolled in hospice during the last three days of life (deaths 2003-07)
* Percent of cancer patients seeing ten or more physicians during the last six months of life (deaths 2003-07)

The layout of this specific dataset made it difficult to be loaded on a python dataframe, thus some actions made in order to simplify its layout.

The anomaly of this dataset was that its header was split in two lines as shown on the screenshot below.



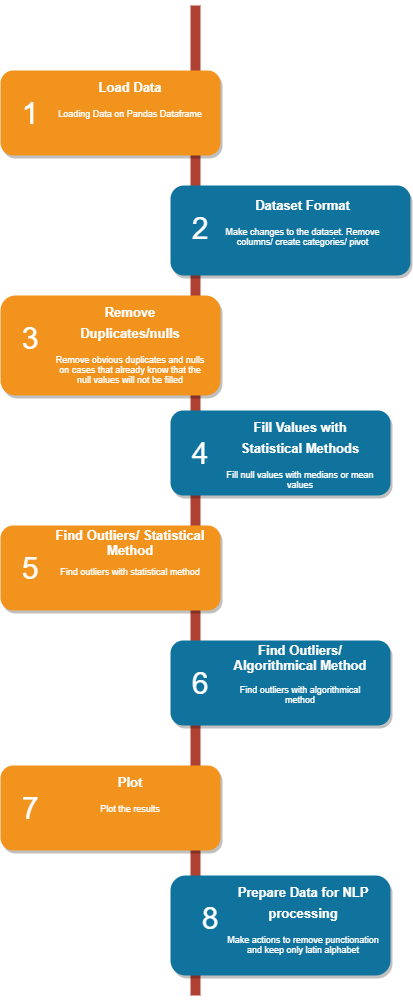
Thus the split lines merged into by keeping the whole information into each cell. As a consequence the dataset loaded successfully on python data frame.

## 4.3 Overall Pipeline

For the needs of data cleaning the data should go throw a data pipeline, namely a series of steps that the data moves through and the output of one step in the process becomes the input of the next.

The steps of a data cleaning process are based on various factors, such as the type of data, the quality of the data, the accuracy of the expected etc. Also a big part of the pipeline is based on the assumptions that been made from the development team in collaboration with the business users who will analyzed the data.

The steps which followed in this specific project are described on the diagram below:



As shown on the diagram firstly the data are loaded on Pandas Dataframe, later on some actions are made in order to eliminate duplicate values, perform data validation, fill or exclude null values, find outliers in the dataset and plot the results. On the next chapters those actions will be examined in detail.

## 4.4 Load Data

The first action is being made when there is the need to work with data which are in a raw format (e.g: excel file, comma separated value file) is to load them on a structure or a database which offers the ability to make actions on a dataset, such as aggregation, filtering etc.

As already has been mentioned for the scope of this project the programming language used for the data manipulation is the Python and consequently the Python Dataframes.

The [Pandas library documentation](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html) defines a DataFrame as a “two-dimensional, size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns)”. In plain terms, the DataFrame can be described as a table of data, i.e. a single set of formatted two-dimensional data, with the following characteristics:

* There can be multiple rows and columns in the data.
* Each row represents a sample of data,
* Each column contains a different variable that describes the samples (rows).
* The data in every column is usually the same type of data – e.g. numbers, strings, dates.
* Usually, unlike an excel data set, DataFrames avoid having missing values, and there are no gaps and empty values between rows or columns.

For the scope of this research the data actually loaded on a Python Dataframe in order to apply action and achieve the expected result.

Creating DataFrames from CSV (comma-separated value)  or XLSX (Microsoft Excel) files is a task in Python code which is executed with specific commands of Pandas Library such as:

* [read\_csv](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html)()
* Read\_excel()

Moreover a useful information regarding these functions is that the data can be loaded either from a file on the local pc or from a file located on a web site. The great advantage of the ability of Python to read remotely located files, is that it makes easier the process of co-working and co-operation.

When data are loaded on a Pandas Dataframe the process will continue by applying the actions needed regarding cleaning on the dataframe.

## 4.5 Dataset Format

After loading the data it is essential to do some actions to the dataset which will make the process of cleaning easier and more efficient. There is no specific term for this kind of actions but those can be described as the ‘formatting of the dataset’.

Regarding the dataset of the current research, the formatting process can be described in bullets as follows:

* Excluding columns not needed in the process of data cleaning.
* Renaming columns either by filling gaps in column titles with underscores or by giving more meaningful names.
* Pivoting the dataset in order to transform the rows to columns and work more efficiently on aggregations.

Beginning with the exclusion of columns, it should be mentioned that it is a mandatory step, because carrying extra columns that will not be useful for the data analysis is not a best practice performance wise. Also the extra columns increase the complexity of the code for the developer because such as in cases of grouping.

The practice of columns’ renaming is also used in order to keep more meaningful or shorter names in the dataset. This method also helps the developer to have a better view on the result or writing code faster because of shorter names.

Regarding the data pivoting, it is a common process when the current layout of the dataset makes it difficult or even impossible to extract the expected results from the dataset.

## 4.6 Remove Duplicates/Nulls

Duplicates records are data points that are repeated in the dataset and they are a common issue in raw datasets.

How duplicate records occur:

* Data are combined from different sources
* The user may hit submit button twice thinking the form wasn’t actually submitted.
* A request to online booking was submitted twice correcting wrong information that was entered accidentally in the first time.

Consequently duplicate records undoubtedly should be confronted as false records and for the scope of this project they were removed. Python offers a specific command for duplicates eliminiation:

* .drop\_duplicates()

Null values or missing values is another common issue when the data analyst comes up with a dataset.

How null values occur:

* The application user did not fill a field by mistake
* During a research the respondent did not answer to a question
* Bug in the application

Most of the times null values should be removed because they cannot contribute in the data analysis. Also null values may affect the way that the python code works e.g.: they may not let the developer to perform some actions during coding.

As implied, for the scope of the project several null values removed.

Nonetheless, under certain circumstances null values should be kept or filled and this will be examined on the next chapter of the project.

To conclude, it should be mentioned that regarding applications and databases the developers may enforce constraints such as uniqueness of record or not allowing nulls in records these kinds of practices eliminate the presence of duplicate records and nulls.

## 4.7 Fill Null Values

As already mentioned in certain occasions there is the need to keep and/or fill nulls to perform data analysis, because removing null values may lead to loss of information and data or it may distorts the results if the percentage of missing values is high compared to the whole dataset.

For the scope of the current project, decided that some null values should be replace instead of them been dropped.

There are plenty of methods that perform the filling of null values thus the developer should choose the one that fits best regarding the quality of the results, the performance and the ease of use.

One group of methods that is very close together is the method of replacing null values with Mean, Median or Mode. This strategy can be applied on a feature which has numeric data like ‘Persons Living with HIV/AIDS Rate’.

By using the group of methods mentioned above, actually there is a calculation of mean, median or mode of the feature and the result of the calculation replaces the missing values. This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns. Replacing with the above three approximations are a statistical approach of handling the missing values.

For the needs of the projects the method used for replacing null values was the median which is provided by Python with the following command:

* fillna(x.median())

## 4.8 Outlier Detection

In data analysis, outlier detection is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Typically the outlier detection and removing is a common technique for data analysis purposes.

There are plenty of methods that the developer can go through and perform outlier detection to the dataset. Among them there are two parent groups/methods that contain many techniques of outlier detection and those are the Statistical Method and the Algorithmic Method.

For the scope of the project both of the methods were used in order to compare their performance and results in detail. Each of them has pros and cons that will be examined in detail on the upcoming sub-chapters.

### 4.8.1 Statistical Method

As already mentioned, under the umbrella of statistical methods for outlier detection there are a lot of methods. For the scope of this project, the Z-score method was selected to detect the outliers of the dataset.

The Z-score is the signed number of standard deviations by which the value of an observation or data point is above the mean value of what is being observed or measured.

The way that Z-score works, is to describe any data point by finding their relationship with the Standard Deviation and Mean of the group of data points. Z-score is finding the distribution of data where mean is 0 and standard deviation is 1 i.e. normal distribution.

To be more precise, while calculating the Z-score, the data are rescaled and centered. Consequently the data points which are too far from zero will be treated as the outliers. In most of the cases a threshold of 3 or -3 is used i.e if the Z-score value is greater than or less than 3 or -3 respectively, that data point will be identified as outliers.

It is a fact that z-score can be calculated by hand, i.e. translate the mathematical calculations described above in code. But in terms of Python there are automated solutions. During the development of the project, made use of the scipy library and especially the stats module which help the developer to calculate z-score with one command:

* bigCts\_dups = bigCts\_dups[np.abs(stats.zscore(bigCts\_dups['Value']))<1]

After finding the outlier records those can be easily removed by the dataset and proceed to next steps.

### 4.8.2 Algorithmic Method

Despite the fact that statistical methods are easy to use, efficient and effective, the algorithmic outlier detections methods should also been taken into consideration. One of the most common practices regarding algorithmic outlier detection is the use of k – Nearest Neighbors (kNN) algorithm.

kNN is a supervised ML algorithm frequently used for classification problems (sometimes regression problems as well) in data science. It is one of the simplest yet widely used algorithms with good use cases such as building recommender systems, face detection applications etc.

The fundamental assumption in the nearest-neighbor family is that similar observations are in proximity to each other and outliers are usually lonely observations, staying farther from the cluster of similar observations. The image below offers a better understanding of kNN usage.



For the implementation of kNN algorithm on the project, made use of *sklearn.neighbors* library and the *NearestNeighbors* module.

By finding the lonely observations those are considered as outliers and can be removed from the dataset and proceed to next steps.

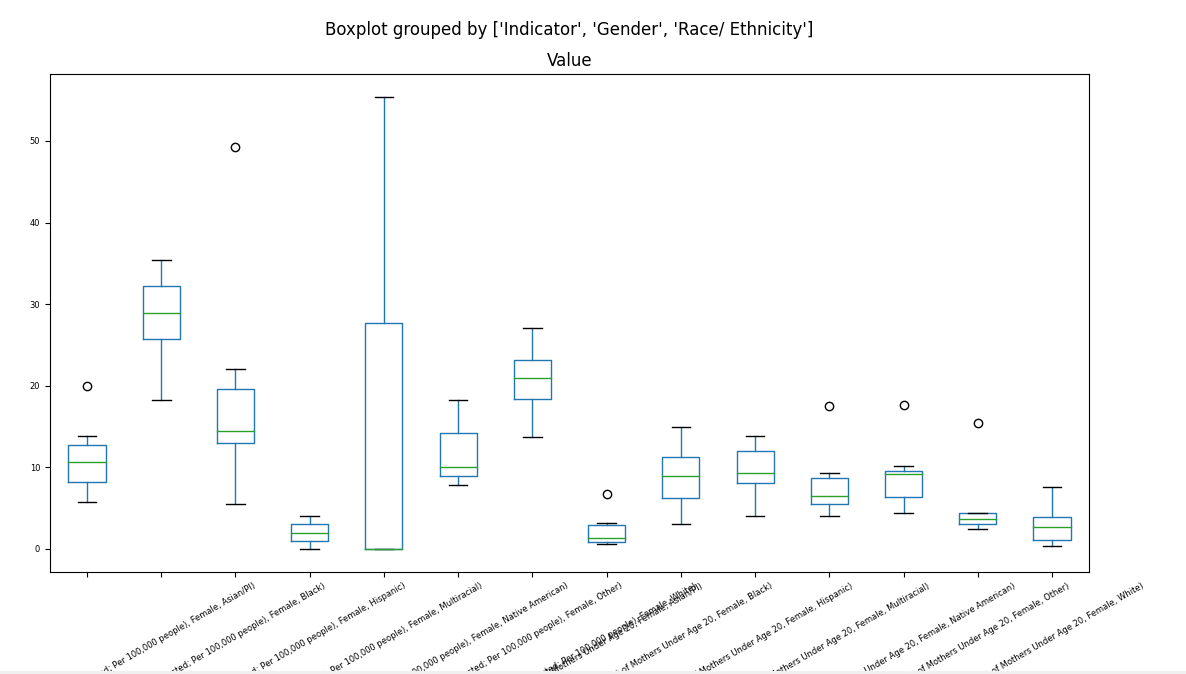
## 4.9 Plots

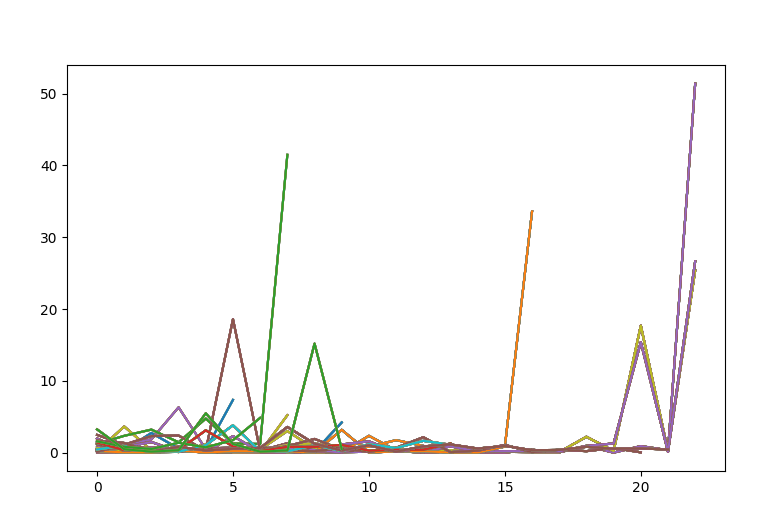
Plots are a very useful tool on the hands of data analyst during the whole data cleaning pipeline. To begin with, the data analyst should choose among plenty of plot types depending on the step being and the decision want to take by seeing the plot. Some of the most frequently used plot types are: line plot, scatter plot, area plot, bar chart, piechart, histogram, kernel density function, boxplot, and scatter matrix plot.

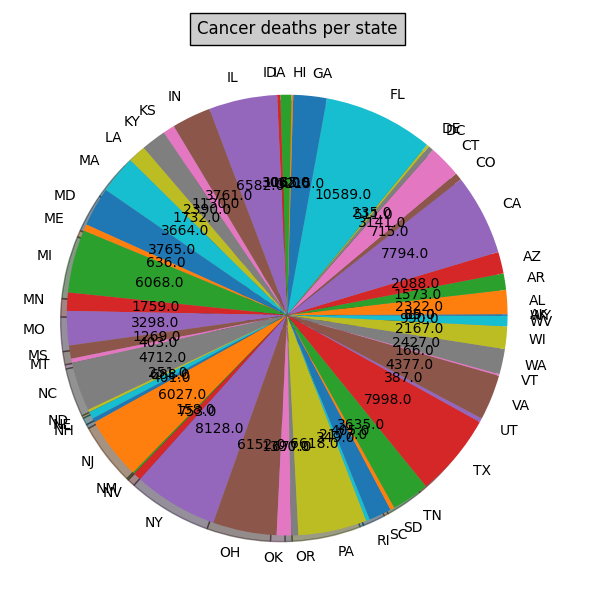
For the needs of the thesis the use of plots made mainly for outliers definition and plotting results.

Regarding the Z-score decision line plots were used in order to define the accurate Z-score that used to remove the outliers values.

Regarding the result made use of box plot and pie chart. The plots used in the thesis, are offered by *matplotlib* library, *pyplot* module. A sample of the plots produced by the code, are displayed below:







## 4.10 Transform Data for NLP Processing

NLP stands for Natural Language Processing, namely it is broadly defined as the automatic manipulation of natural language, like speech and text, by software. The preparation of the data for NLP was made as a case study to examine the performance of some common techniques and how efficient they are.

Preparation of data for NLP should confronted as a purely data cleaning process regarding only the text and not the data as a whole. NLP software demands to be fed with text of human speech. Although in raw data a lot of non-human speech elements can be found.

Elements like:

* URL addresses
* Signs
* Special Characters
* Emails

are not appropriate data to be fed on NLP software, consequently they have to be removed.

In order to remove all the inappropriate elements of NLP, made use of *NLTK* (Natural Language Toolkit) library. Through some research and implementation it turns out that this is a very powerful toolkit to remove any extra character except natural language.

## 4.11 Evaluation/Outcomes

The pipeline of this project covers a well-rounded data cleaning process that could be also applied in real life business scenarios. To recap, the current project’s scope was to clean two different datasets by using two different manners. From this process the outcome should be compared regarding which data cleaning method performs better and produce more accurate results.

The table below show the performance of each method on each dataset:

|  |  |  |
| --- | --- | --- |
| **Dataset Name** | **Outlier Detection Method** | **Result in seconds** |
| United States Big Cities Data | Statistical (Z-Score) | 0.057701111 |
| Algorithmic (kNN) | 1.386090279 |
| United States Hospitals - Cancer Events | Statistical (Z-Score) | 0.045973539 |
| Algorithmic (kNN) | 45.58999705 |

Regarding the amount of time needed by each method, it is occurs from the results that statistical method performs significantly better than the algorithmic. The main reason that this happens is that in order to apply the kNN algorithm in these types of dataset it was mandatory to use iteration over the dataset. More precisely the *.iterrows* function was used to perform the iterations. This function is considered as a bad practice in python coding, but it was the best solution to produce the results.

Another fact is that the algorithmic process is heavier than the statistical, consequently even with the absence of iteration factor it would be expected to be slower than the statistical.

Moreover it is noticed that the algorithmic process of the second dataset i.e. United States Hospitals is much slower than the statistical, around 1000 times. That occurs because of the amount of data that the second dataset has. Thus it turns out that the amount of data affect exponentially the iteration usage.

Regarding the quality of data it turns out once again that the statistical method produced more accurate data, namely predicted better the outlier values.

To conclude during the implementation, the statistical method turns out that produced more accurate and understandable results and it would be recommended for further investigation in static datasets.

# 5. Conclusion

This study on data cleaning presented a holistic approach of cleaning datasets and mainly spot outliers by two different, commonly used, methods.

Hopefully this research contributed to an understanding of how different algorithms affect drastically the performance in Python code and the reader will be suspected about which method will use in similar tasks in the future.