

**Evaluation of Data Cleaning methods in Healthcare Domain**

Student: Nikolaos Moutsopoulos

Ε-10102

Supervisor: Dimosthenis Kyriazis

**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)

**List of Tables**

[Table 4.1 Results regarding running time 32](#_Toc73033223)

**List of Figures**

[Figure 2.1 Data Mining Steps 6](#_Toc73033249)

[Figure 4.1 Caption of United States Big Cities Dataset 18](#_Toc73033250)

[Figure 4.2 Caption of U.S Hospitals - Cancer Events 20](#_Toc73033251)

[Figure 4.3 Project Pipeline 22](#_Toc73033252)

[Figure 4.4 Graphical example - Find outliers with kNN algorithm 28](#_Toc73033253)

[Figure 4.5 Boxplot - helps to find outliers 29](#_Toc73033254)

[Figure 4.6 Lineplot - helps to find outliers 30](#_Toc73033255)

[Figure 4.7 Piechart - helps to report aggregated data 31](#_Toc73033256)

# 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. These data, especially when related to healthcare, cannot be wrong, inaccurate, incomplete or unrecognizable to the operations and processes that consume them. The consequences 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 thesis 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.

# Data Mining

## 2.1 Introduction

Nowadays there is a great competition among companies because of the globalization and the very fast pace of production of new products and inventions. Every company is trying to thrive in their respective sector in order to expand, success and avoid bankruptcy. In order to achieve that the companies should take effective and fast decisions. Every CEO, manager or decision maker needs data in order to decide the next step or the next goal, either it is an e-shop based business, a clothing store or a software house. Here comes data mining, the process of collecting data and produce meaningful results to help the key persons to take the right decision.

Data mining is the process of collecting raw data from various sources, such as the users’ logs of a web application, transform them in order to be digested in a database and apply techniques to produce meaningful results to the respective business or organization. Usually the people who get involved in this process are data scientists, data analysts, business intelligence engineers etc.

The process of data exploitation is almost mandatory in some of the biggest business sectors of the current era. For example the banks are obliged to report their risk factor or they are obliged to detect fraud events. In order to achieve those goals they have to process large amount of data and produce reports instantly. Also all the well – know email platforms use data mining techniques in order to identify spam Emails.

Data mining is based on some basic techniques to produce results. Those techniques can be combined or used individually depending on the data input and the desired outcome that the developers want to achieve. Some of the techniques used in a data mining process are *Association, Classification, Clustering* etc. which are going to be examined in detail later in the thesis project.

The events examined in data mining process 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.

Despite the beneficial aspect of data mining there are also some disadvantages on this concept. One of the most common problems is the data quality. When the data are collected from sources where the human factor is present among them, then a high percentage of records can be outliers or useless for data analysis. Moreover the security and data privacy concerns both the users and the data collectors. On the one side the users are afraid of being exposed if their data leak, on the other side the companies must comply with the rules and laws of data privacy in order to avoid to be issued in case of data leakage.

## 2.2 Steps of Data Mining

As already mentioned nowadays organizations have to their disposal a lot of sources of data, such as logs, sales data, website visitors data, and more every day. Because of the need for those data to be exploited, have been defined some phases/ steps in order to build a data mining pipeline. Data mining is a process of separate steps which may be repeated through the pipeline of the data. The decision about the need of a step repetition depends on the previous step outcome. Those steps have been defined by the Cross-Industry Standard Process for Data Mining (CRISP-DM) which it is assumed as a guideline to develop the data mining process

The 6 CRISP-DM phases

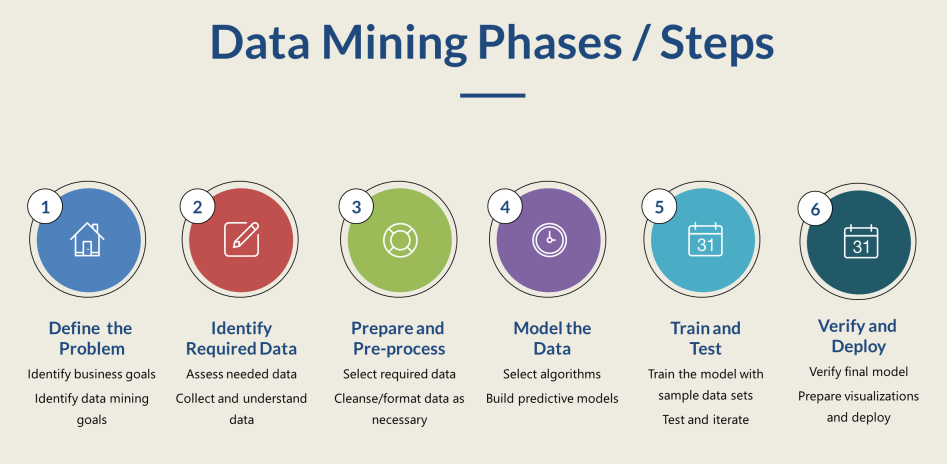


Figure 2.1 Data Mining Steps

The CRISP-DM comprises a six-phase workflow. It is a flexible process and the people who are based to develop their data mining pipelines through this process are encouraged to be agile and go back and forth through those steps depending of the outcome of the respective step.

**1. Define the problem**

This step could be considered as the analysis phase of the project. The development team is called to understand the business requirements and the project goal in order to translate it to a data mining problem definition. This process requires the contribution of both the development team and the business department which will receive the results of the data mining project to use it on their benefit. At the end of this phase and before the start of phase two it is almost mandatory to be written clearly the data that will be produced as outcome to the business department in order to be avoided any misunderstanings. (Bharati M. Ramageri et al., 2013)

**2. Identify Required Data**

During this step the development team is called to identify the data sources and the data that will be used in the next steps. Also they should dive into those data in order to evaluate the quality of them and if all of the data which needed to produce the outcome are at their disposal. The fact is that the data may come from multiple sources and the development team should investigate and analyze the type of data from each source in order to be ready to load them in their data warehouse. At the end of this process the data team has define accurately the data that will be used during the project. (Bharati M. Ramageri et al., 2013)

**3. Prepare and Pre-process**

On this step it takes place the development which transforms the raw data to a final dataset on database tables with the specific information that will be used to the next steps. The outcome of this step can be examined in collaboration with the stakeholders and the business department in order to be identified the dimensions and variables that will be used on the model creation. (Bharati M. Ramageri et al., 2013)

**4. Model the Data**

In this phase, will be selected the modeling techniques that will be applied on the given data and produce the desired outcome. The techniques that can be used by the data mining team are clustering, predictive models, classification, estimation, or a combination. It may be mandatory to return to prepare and pre-process phase if the chosen modeling technique needs different input (e.g. variables and dimensions). (Bharati M. Ramageri et al., 2013)

**5. Train and Test**

When the modeling creation is completed the outcome should be tested and examined if it produce the desired outcome as it is has been described on the first step of data mining process. If there is declination of the outcome and the expected result then some of the previous steps should be analyzed again and a solution must be found. The model has been created may answer requirements that have not been asked. In this situation the outcome should be discussed with the stakeholders and be decided about the solutions. This step in development terms is called as Integration Test or UAT (User Authentication Test). (Bharati M. Ramageri et al., 2013)

**6. Verify and Deploy**

This final step call the data mining team to produce an outcome visible to those who need the report and take the decisions of the organization (e.g. stakeholders, business department). The deployment maybe an easy process like producing a single report or a more complex procedure like engage the whole data mining process to an existing data flow. (Bharati M. Ramageri et al., 2013)

## 2.3 Data Mining techniques

Data mining developers can choose among multiple methods and techniques in order to solve the business problems and produce the expected results. On this section will be examined in detail some of the most common methods being used in the process of data mining process.

**Association**

In Association the goal is to find similarities between different observations on the data to predict patterns. This technique is commonly applied to build recommendation systems. For example when the customer purchases a road bicycle(A), then they also buy a bicycle leggings (B) and this occurs in 55% of the observations. This pattern occurs in 8.2% of bicycle purchases. An association rule in this situation can be “A implies B, where confidence factor is 55% and the support factor is 8.2% is the support factor”. By creating these type of rules a website can suggest to his customers relevant product to buy when they buy a bicycle, or the organizations can send newsletters with personalized offers to buyers that need certain products. (David L. Olson et al., 2013)

**Classification**

Among the most common techniques used in data mining is classification. Classification is to identify an index or a variable of the given data and map each item of the data into separate classes. This process requires to know the predefined classes and the number of attributes and a training set of data. Then the classification algorithm automatically divides the data to the respective group. To achieve this process certain mathematical techniques are used such as neural networks, statistics, linear programming etc. An example to understand the outcome of this process is that through classification can be identified the age group of a website visitors and create aggregates about which age group make the more visits. (David L. Olson et al., 2013). In order to implement classification techniques the developer has in his disposal some well known algorithms which perform this job. Some of those algorithms are explained in detail below:

* Decision Tree

A decision tree is a popular classification method. Divide and conquer is the methodology used in this process. The scope is to break down the data in categories. The decision tree's structure is organized so that it contains the root, which is the tree's topmost node, branches, which are internal nodes, and leaf nodes, which are not further classified. Consequently, the leafs are the classes produced by the algorithm. The advantages of this process is that it is easy to comprehend and visualize, requires minimal data preparation, and can handle both numerical and categorical data. The disadvantages is that Decision Trees can produce complicated trees that are difficult to generalize, and they can be unstable since slight changes in the data might result in the generation of an entirely different tree.

* Logistic Regression

The classification algorithm Logistic Regression is used to assign observations to a discrete set of classes. Classify Emails to spam or not spam is an example of classification issues. The logistic sigmoid function translates the output of logistic regression into a probability value. Regarding its advantages Logistic regression was created specifically for classification, and it's especially good for figuring out how numerous independent factors affect a single outcome variable. On the other side it only works when the predicted variable is binary, assumes all predictors are independent of one another, and assumes no missing values in the data.

* Naïve Bayes

Bayes theorem is the foundation for the naive bayes classifier. For the provided set of classes, a hypothesis is constructed. The independence assumption is made in the Naive Bayes algorithm. The values of the attribute are chosen based on the desired value, and they are independent of one another. The Naïve Bayes classifier uses a simple approach. It is possible to classify the supplied cases with a modest quantity of training data.

For example, based on the color orange and the shape round, the fruit is identified as an orange fruit, demonstrating that it is an independent model. This strategy can also be used in more complicated circumstances.

* k-NN algorithm

Neighbor-based categorization is a type of lazy learning because it doesn't try to build a general internal model and instead just stores instances of the training data. The classification is determined by a simple majority vote of each point's k nearest neighbors. k-NN algorithm is available by **Scikit-learn** and has been used in order to perform outlier detection in the current thesis. This technique is straightforward to use, robust to noisy training data, and successful when dealing with large training data. The fundamental downside of KNN is that it becomes much slower as the volume of input grows, making it an unsuitable solution in situations when rapid predictions are required.

**Clustering**

The cluster methods take raw data and through specific algorithms divide those data into certain groups. The methods used in order to cluster data are pretty similar to those that perform classification. Although exist some very powerful algorithms that helps the developers to perform clustering in an efficient way such as k Nearest Neighbors (k-NN) algorithm. The k-NN algorithm is a supervised machine learning algorithm that can be used mainly for classification problems but it performs well also in regression predictive problems. (<https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm>). It is a fact that some of the algorithms in order to perform classification can also be used in order to implement clustering. However below examined some of the algorithms in detail which are used for clustering:

* K-means

With respect to the clustering error, the k-means algorithm produces locally optimal solutions. It's a rapid iterative approach that's been applied to a variety of clustering applications. It's a point-based clustering algorithm that starts with the cluster centers at random positions and moves them at each stage to reduce clustering error. The method's biggest drawback is its sensitivity to the cluster centers' initial placements. As a result, numerous runs with different initial positions of the cluster center must be scheduled in order to find near optimal solutions using the k-means method. (Aristidis Likas et al., 2012).

* DBSCAN

Density-Based Spatial Clustering of Applications with Noise. Finds core samples of high density and expands clusters from them. This algorithm finds a good appliance for data which contains clusters of similar density (<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>). The DBSCAN algorithm uses two parameters minPts and eps (ε). The minPts is the minimum number of points clustered together for a region to be considered dense. While eps (ε) is a distance measure that will be used to locate the points in the neighborhood of any point.

**Prediciton**

Prediction analysis can be achieved by using regression techniques. The fundamental purpose is to discover the behavior between two variables and how the one interacts to the other. For example the sales and the profit are two variables and the business department would like to predict what will be the profit at the end of the year depending to sales. Here comes the appliance of regression algorithms on historical data from both variables that creates a model. Through this model can be predicted the profit regarding the sales of the running year. (David L. Olson et al., 2013)

**Sequential Pattern**

This method analyze the data in order to find similarities in data during a certain period of time. For example the visits of an e-shop increasing the first days of a month because the customers get their payroll. These observations can be used by stake holders to identify relationships among data and predict future trends. This approach can be applied on databases that have time-series characteristics. (David L. Olson et al., 2013)

**Outlier Detection**

The method of outlier detection targets on specific variables and tries to find items that are very different from the majority of the rest items. For instance if the observations refer to ages of people then if the algorithm spots that a person has 180 years of age then this observation may be considered as an outlier. To perform this type of detection are being used various mathematical process and some of them are being used for the purposes of this project in order to perform data cleaning.

# Data Cleaning

## 3.1 Introduction

As already has been examined the whole process of data cleaning it should be mentioned that data cleaning is considered to takes places at the third phase of data mining (i.e.: Prepare and pre-process data). Data cleaning is the process of identifying and correcting data that is corrupted, duplicated, missing, or erroneous. These activities and processes must be ingrained in everyday operations for the organization to operate efficiently and make accurate judgments that lead to beneficial outcomes. Data cleansing has been crucial in assuring the quality of data for enterprise applications. Many data cleaning algorithms have been converted into tools to detect and maybe correct particular classes of errors such as outliers, duplicates, missing values, and violations of integrity constraints, therefore there has been a lot of research in this field. (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 Approaches

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 Healthcare Domain

Healthcare is a domain in which data have a great value and their usage may give very beneficial results 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.

Organization which belong to the health care sector collect data from various sources which are being analyzed and later they share the produced results. The produced results may be protected health care information (PHI), thus they must take seriously the subject of security and the data that they expose. In order to ensure the data integrity, portability and accessibility for such type of information the organizations should invest a lot of effort in data cleaning.

Like the rest organizations who perform data mining, the organizations store data in large databases and/or data warehouses. These could be linked to each person’s electronic health record EHR, decision support system, revenue cycle management, and a variety of other apps that help the healthcare ecosystem function together more effectively. The usefulness of healthcare big data is enormous, as it aids in improving care, increasing revenue, and facilitating improved decision-making. With the existence of dirty data the previous statement is infeasible. Thus organization can make their data more valuable by using data cleansing procedures during the pipeline of data mining. When integrating various data streams, consistency is essential and when filthy data is merged, it loses its capacity to be actionable.

All healthcare information systems (HIS) would work in perfect harmony under ideal circumstances. Field matching, as well as duplicates and other inconsistencies, would not be a hindrance. That, unfortunately, is not the case. There is currently no established method for data cleaning in the health care domain, thus the development teams must put effort in order to make the respective data cleaning.

However, migrating data from one system to another or fast aggregating multiple data sets and automatically having a working process are still difficult. It is observed how difficult it is to translate data from one system to another, even when they are in the same category, not only in health care but in every organization which should use data cleaning techniques. As a consequence if data are migrating from one EHR to another with ease, integrating data outputs or shifting information into a whole different sort of platform becomes quite difficult.

Dirty data is the result of a combination of variables, some of which are more significant than others. Duplication is one of the most serious difficulties. Duplicate records make up 5-10% of a hospital's EHR, according to study. For healthcare organizations with many locations, that percentage rises to 20%. (<https://infowerks.com/healthcare-data-cleaning/>).

Duplications can occur for a variety of reasons, including mistakes in spelling or other patient information. As additional patients are added, the system's characteristics may prevent it from searching for duplicates. The fact that filthy data is incomplete is another sign. Records may be useless if they lack all of the necessary fields. A patient record list that omits prior diseases or allergies is not only incomplete, but it may also have an impact on care. User error or system constraints can also lead to incomplete data.

Inaccuracies are the third major source of filthy data. Errors in the initial set-up (misspelled names, transposed numbers, etc.) may have occurred, or the data may not have been updated appropriately. It's difficult to engage with patients and harness your data for better results and insights if the data source for them is not precise, from contact information to insurance codes.

When the data quality is poor there are many consequences to organizations and the result is that the latter lose a lot of money each year. The amount of monetary losses obviously depends on the size of the organization but for reference the losses may equates to [$9.7 to $14.2 million](https://www.gartner.com/smarterwithgartner/how-to-stop-data-quality-undermining-your-business/" \t "_blank). (<https://infowerks.com/healthcare-data-cleaning/>). Those losses may occur from various reasons such as unpaid reimbursements from payers. More over the losses may be not only monetary, for example the organization may lose time to integrate data from old to new platforms or missing insights which would help to identify cost cutting procedures.

# 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.



Figure 4.1 Caption of United States Big Cities Dataset

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.

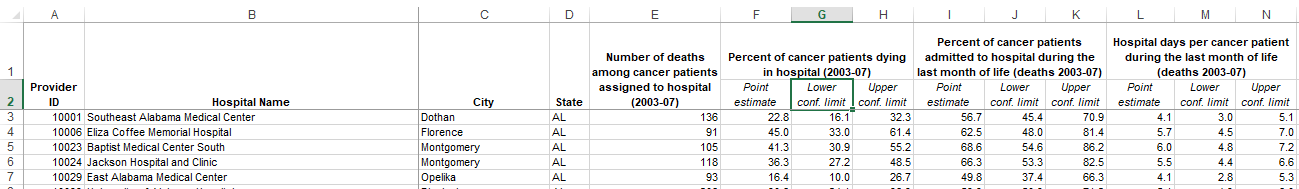


Figure 4.2 Caption of U.S Hospitals - Cancer Events

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:

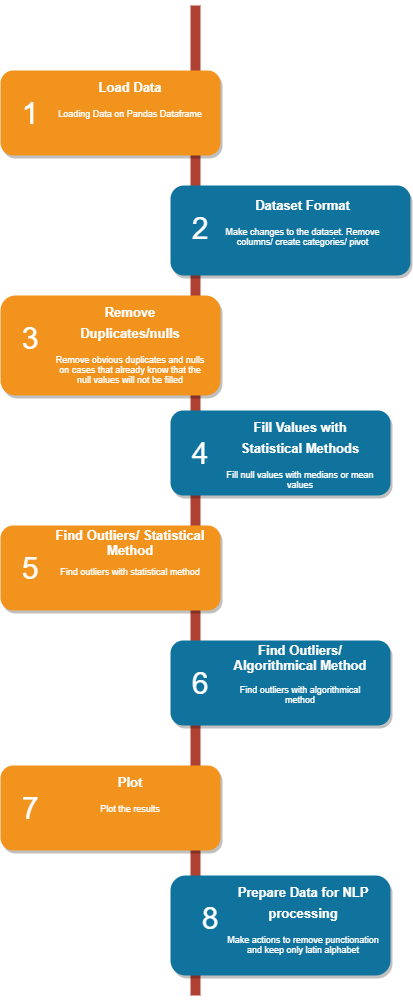


Figure 4.3 Project Pipeline

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.

The Python code used for the implementation of the data load is appended below:

bigCts = pd.read\_csv("https://raw.githubusercontent.com/surping/thesis/master/Big\_Cities\_Health\_Data\_Inventory.csv")

bigCts = pd.DataFrame(bigCts,columns=['Indicator Category', 'Indicator', 'Year', 'Gender', 'Race/ Ethnicity', 'Place', 'Value', 'Source', 'Methods'])

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.

The Python code used for the implementation of the dataset format is appended below:

bigCts['Indicator'] = bigCts['Indicator'].replace(['Opioid-Related Mortality Rate (Age-Adjusted; Per 100,000 people) \*These data should not be compared across cities as they have different definitions.'],'Opioid-Related Mortality Rate (Age-Adjusted; Per 100,000 people) \*These data should not be compared across cities as they have different definitions')

bigCts\_null\_cols = pd.DataFrame(bigCts,columns=['Indicator Category','Indicator','Gender','Place','Value'])

DapHospPvt = DapHosp.melt(id\_vars=['Provider\_ID', 'Hospital Name', 'City', 'State'],

        var\_name='Category',

        value\_name='Value')

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.

The Python code used for the implementation of the duplicate removal and negative values removal is appended below:

DapHosp = DapHosp.drop\_duplicates()

DapHosp = DapHosp.fillna(1000000)

DapHosp[DapHosp.iloc[:, 4:38]  < 0] = np.nan

DapHosp = DapHosp.fillna(DapHosp.median())

DapHosp[DapHosp == 1000000] = np.nan

DapHosp = DapHosp.iloc[:, :]

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())

The Python code used for the implementation of the fill of Null Values is appended below:

bigCts\_toFill.Value\_x = bigCts\_toFill.groupby(['Indicator Category','Indicator','Gender','Place'])['Value\_x'].apply(lambda x: x.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.



Figure 4.4 Graphical example - Find outliers with kNN algorithm

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:

* In the box-plot displayed below are pictured the categories of the big Cities dataset and the distribution of the data per Category. With the blank bullets which are places above the boxes are displayed the outlier values for the respective category.

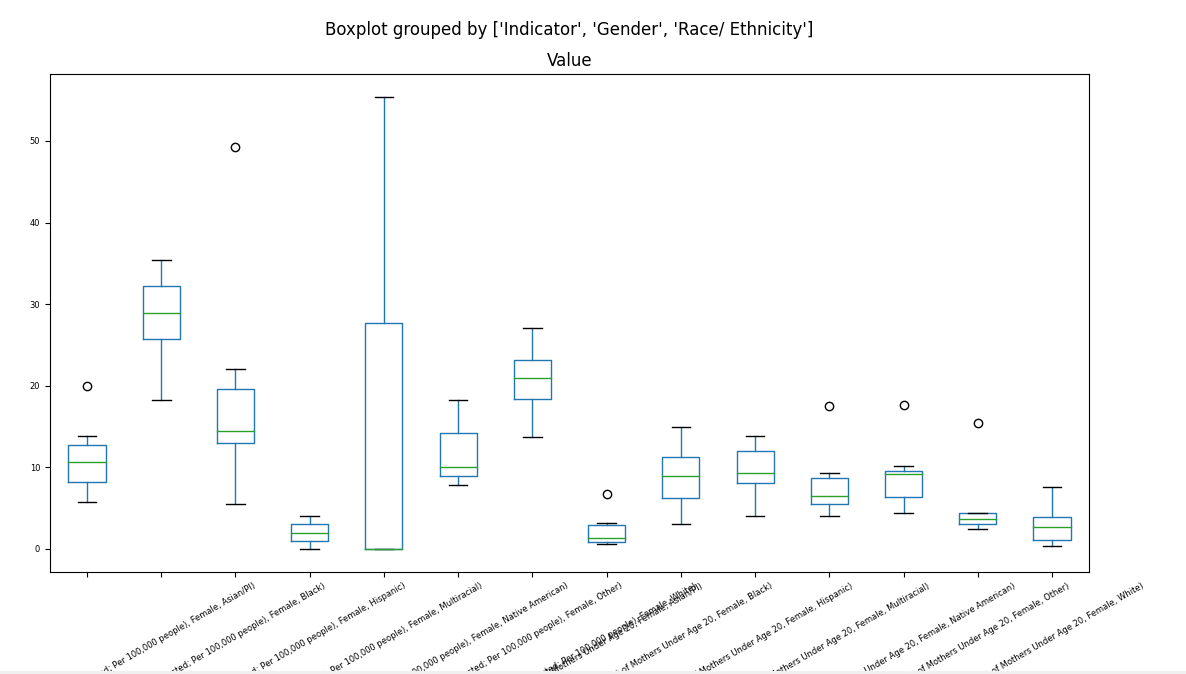


Figure 4.5 Boxplot - helps to find outliers

* The line plot displayed below is the result of the appliance of the k-NN algorithm on the Big Cities Dataset, which helps to spot the lines (categories) which are not correlated to the rest of the dataset. In this occasion the lines which are too high from x-axis give a perspective of the categories of the dataset which contain outliers.

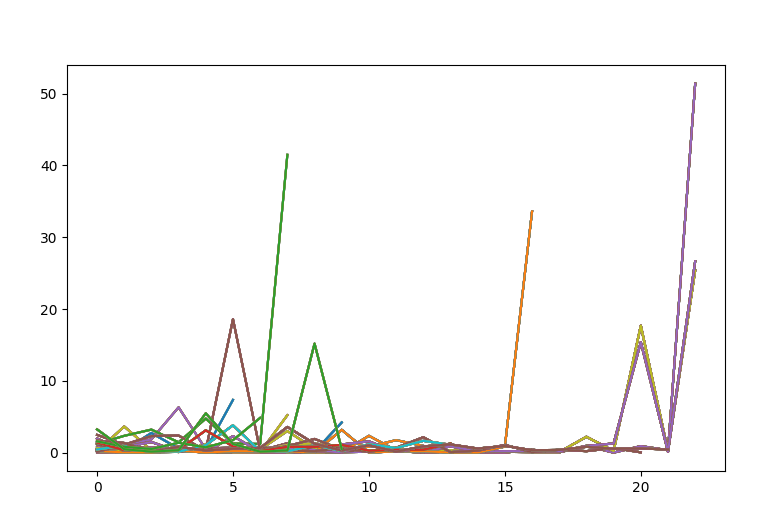


Figure 4.6 Lineplot - helps to find outliers

* The piechart displayed below pictures the Cancer Deaths per state after the data cleaning process which occurred from the Cancer Events dataset.

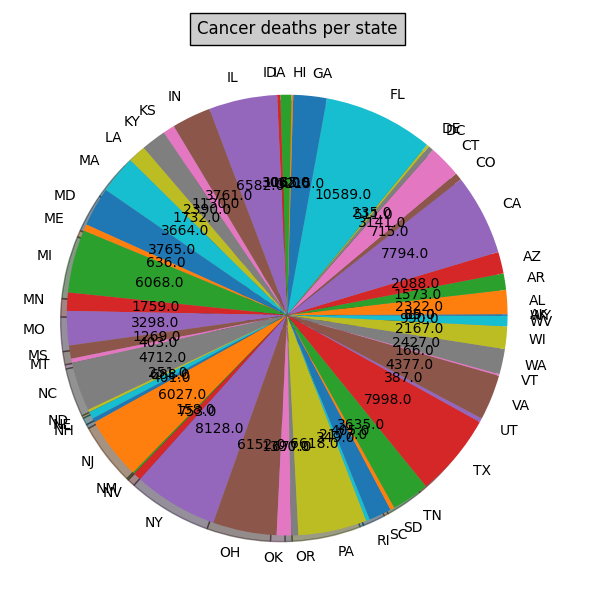


Figure 4.7 Piechart - helps to report aggregated data

## 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:

Table 4.1 Results regarding running time

|  |  |  |
| --- | --- | --- |
| **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.

# Conclusion

In this paper conducted a comparative analysis and evaluation of two different data cleaning methods, algorithmic and statistical, on data which occurred from health care sector, in order to indicate which method is more efficient regarding performance and the quality of the produced result. During the implementation of both methods took place the use of techniques such as de-duplication or other functionalities provided naturally by python language or the extension libraries such as pandas, numpy, seaborn etc. Also made use of charts which helped in the rights decisions during coding or displayed the expected result. Regarding the data cleaning approach with statistical method, made use of mathematical functions like median and z-score, while on the other hand, for the purposes of algorithmical approach made use of k Nearest Neighbors algorithm. Moreover on the one dataset which contained columns with free text, implemented and approach of natural language processing so as to be ready to imported on an algorithm which may conduct further analysis on the free text.

This comparison on using the same algorithms on two different datasets for data cleaning helped to evaluate better the outcome of the experiment. Both implementations ended up that the usage of statistical methods instead of using more complex algorithms (i.e. k-NN) for data cleaning were much more efficient from every aspect, namely the quality of the results was more accurate, the performance of the code during running was faster and the coding implementation was simpler for the developer. Another key thing to mention is that the preparation of free text in order to transform it to NLP-ready text is a heavy procedure that consumes a lot of time during code running.

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. To this end, there is a lot of additional research to perform regarding the search of the best implementation for data cleaning purposes and undoubtedly it a parametrical process, namely the procedure will be chosen depending on the type of data, the infrastructure and the desired result.

# References

Abedjan Z, Chu X, Deng D, Castro Fernandez R, Ilyas I. F, Ouzzani M, Papotti P, Stonebraker M, Tang N. (2016). Detecting data errors: Where are we and what needs to be done? In Proceedings of the VLDB Endowment, pp. 993 - 1004

Abedjan Z, Akcora C. G, Ouzzani M, Papotti P, Stonebraker M. (2015). “Temporal Rules Discovery for Web Data Cleaning”, in Proceedings of the VLDB Endowment, pp. 1-13

Angiulli F, Fassetti F. (2007) “Detecting distance-based outliers in streams of data,” in Proc. 16th ACM CIKM, Lisboa, Portugal, pp. 811–820.

Bharati M. Ramageri, Dr. B.L. Desai (2013). “ROLE OF DATA MINING IN RETAIL SECTOR” in International Journal on Computer Science and Engineering (IJCSE), pp.47-48

Birant D, Kut A. (2006). “Spatio-temporal outlier detection in large databases,” J. Comput. Inform. Technol., vol. 14, no. 4, pp. 291–297.

Dasu T, Johnson T, Muthukrishnan S, Shkapenyuk V. (2002). Mining database structure; or, how to build a data quality browser. In Proc. ACM SIGMOD International Conference on Management of Data.

Dasu T, Loh J.M. (2012). Statistical distortion: Consequences of data cleaning. Proceedings of the VLDB Endowment, Vol. 5, No. 11.

David L. Olson, Dursun Delen (2008). Advanced Data Mining Techniques, pp.16-18. <https://www.researchgate.net/profile/David-Olson-21/publication/220695151_Advanced_Data_Mining_Techniques/links/00b4952b07ee813b62000000/Advanced-Data-Mining-Techniques.pdf>

Elmagarmid A. K, Ipeirotis P. G, Verykios V. S (2007). Duplicate Record Detection: A Survey. In IEEE Transactions on knowledge and data engineering, Vol. 19 (1), pp 1 – 16

Franke C, Gertz M. (2009). “ORDEN: Outlier region detection and exploration in sensor networks,” in Proc. ACM SIGMOD, Providence, RI, USA, pp. 1075–1078.

Greenwald M, Khanna S. (2001). Space-efficient online computation of quantile summaries. In Proc. ACM SIGMOD International Conference on Management of Data.

Gupta M, Aggarwal C. C, Han J. (2011). “Finding top-k shortest path distance changes in an evolutionary network,” in Proc. 12th Int. Conf. Adv. SSTD, Minneapolis, MN, USA, 2011, pp. 130–148.

Gupta M, Gao J, Aggarwal C.C, Han J. (2014). Outlier Detection for Temporal Data: A Survey. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 26, NO. 9, pp 2250-2267

Heidari A, McGrath J, Ilyas I. F, Rekatsinas T. (2019). HoloDetect: Few- Shot Learning for Error Detection. In 2019 International Conference on Management of Data (SIGMOD ’19), June 30-July 5, 2019, Amsterdam, Netherlands. ACM, New York, NY, USA, Article 4, 18 pages. https://doi.org/10.1145/3299869.3319888

Hellerstein J. M. (2008). Quantitative Data Cleaning for Large Databases. EECS Computer Science Division UC Berkeley http://db.cs.berkeley.edu/jmh

Hill D. J, Minsker B. S, Amir E. (2007). “Real-time Bayesian anomaly detection for environmental sensor data,” in Proc. 32nd Conf. IAHR In Proc. International Conference on Very Large Data Bases (VLDB).

Ilyas I. F, Chu X. (2015). Trends in cleaning relational data: Consistency and deduplication. Foundations and Trends in Databases, 5(4), pp. 281–393

Kolahi S, Lakshmanan L. (2009). On approximating optimum repairs for functional dependency violations. In ICDT, pages 53–62

Levina E, Bickel P. (2001). The earth mover’s distance is the mallows distance: some insights from statistics. In ICCV, pp 251–256.

Madni, H. A., Anwar, Z., & Shah, M. A. (2017). Data mining techniques and applications — A decade review. 2017 23rd International Conference on Automation and Computing (ICAC).

Manku G. S, Motwani R. (2002). Approximate frequency counts over data streams.

Pipino, L., Lee, Y. W., Wang R. Y. 2002. Data quality assessment. Commun. ACM, 45(4), pp 211–218 Vol. 8, no. 3, pp. 275–300

Prokoshyna N, Szlichta J, Chianf F, Miller R. J, Srivastava D. (2015) . Combining Quantitative and Logical Data Cleaning. Proceedings of the VLDB Endowment, Vol. 9, No. 4, pp 300 - 311

Rahm E, Do H.H. (2000). Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin, 23(4), pp. 3–13

Yamanishi K, Takeuchi J.-I, Williams G, Milne P. (2004). “Online unsupervised outlier detection using finite mixtures with discounting learning algorithms,” Data Mining Knowl. Discov.

Yang D, Rundensteiner E. A, Ward M. O. (2009). “Neighbor-based pattern detection for windows over streaming data,” in Proc. 12th Int. Conf. EDBT, pp. 529–540.

Wu E, Madden S. (2013). Scorpion: explaining away outliers in aggregate queries. Proc. VLDB Endow. 6, 8 (June 2013), pp. 553-564.

Zhang Y, Meratnia N, Havinga P. J. M. (2008). “Outlier detection techniques for wireless sensor networks: A survey,” Centre Telemat. Inform. Technol. Univ. Twente, Enschede, The Netherlands, Tech. Rep. TR-CTIT-08-59

Aristidis Likas, Nikos Vlassis, Jakob Verbeek. The global k-means clustering algorithm. [Technical Report] IAS-UVA-01-02, 2001, pp.12.