POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Exploring Data Preparation Strategies for Data Stream Analysis

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Giovanni Siracusa**

Student ID: 969524
Advisor: Prof. Cinzia Cappiello
Co-advisors: Camilla Sancricca
Academic Year: 2022-23

Abstract

Data streams are gaining increasing importance, particularly in applications like the Internet of Things, Edge computing, and Industry 4.0. Analyzing these data streams accurately necessitates the highest possible data quality. Data retrieved from sensors or other sources may contain faults, errors, or missing values, making it preferable to correct them before applying any analysis tool, such as a machine learning model. This thesis aims to improve an existing framework for preparing data. Currently, the framework uses a knowledge base to recommend the most appropriate data preparation actions after conducting data profiling and data quality assessment. The introduced improvements aim to expand the scope of manageable data sources, specifically incorporating the handling of data streams. The goal is to investigate the adaptability of techniques designed for tabular datasets in the context of data streams. The dynamic nature of data streams requires the implementation of appropriate techniques, such as an incremental, continuous or windowed approach. This will optimise efficiency and decrease latency. The initial step involves incrementally performing data profiling and conducting a windowed data quality assessment, offering a real-time perspective on stream attributes. Subsequently, the knowledge base utilized for suggestions is enriched by integrating techniques specifically tailored for data streams. The selection of these data preparation actions is guided by experimental sessions that assess the impact of outlier detection and data imputation methods on machine learning predictions. The experiments yielded distinctive results, and they were used to analyse the behaviour of data streams at different quality levels.

Keywords: Data Quality, Data Streams, Data Preparation, Data Profiling

Abstract in lingua italiana

I flussi di dati, noti come data streams, rivestono un'importanza sempre crescente, specialmente nei settori dell'Internet of Things, dell'Edge Computing e dell'Industria 4.0. L'analisi accurata di tali flussi richiede che i dati siano caratterizzati da una qualità elevata. I dati raccolti da sensori e da altre fonti possono presentare errori o valori mancanti, pertanto risulta preferibile correggerli prima di sottoporli a qualsiasi strumento di analisi, come ad esempio i modelli di machine learning. Il presente lavoro di tesi si propone di perfezionare un framework preesistente per la preparazione dei dati. Attualmente il framework utilizza una knowledge base per suggerire le azioni più adeguate, in seguito a una valutazione della qualità dei dati. Le migliorie introdotte mirano all'espansione delle fonti dati gestibili, con particolare attenzione allo sviluppo dei data streams. La dinamicità dei data stream richiede l'implementazione di techniche appropriate, utilizzando un approcio incrementale, continuo o a finestre, per ottimizzare l'efficienza e diminuire la latenza. Il primo compito svolto riguarda l'esecuzione incrementale di data profiling e la valutazione della qualità dei dati, che consentono di ottenere una visione in tempo reale degli attributi del flusso di dati. Successivamente, la knowledge base utilizzata per i suggerimenti è stata arricchita mediante l'integrazione di metodi specificamente progettati per i data streams. La selezione di tali metodi è stata orientata da esperimenti volti ad esaminare l'impatto delle tecniche di outlier detection e data imputation sulle previsioni dei modelli di machine learning. Gli esperimenti hanno prodotto risultati peculiari, i quali sono stati utilizzati per analizzare il comportamento dei data stream a diversi livelli di qualità. Tali risultati, tuttavia, sono influenzati dalla scelta del modello di machine learning adottato.

Parole chiave: Qualità dei dati, Flussi di dati, Preparazione dei dati, Data Profiling

Contents

Introduction

Data analysis has emerged as a prominent and influential domain in recent years, captivating the attention of various industries. Recognizing the pivotal role of data, numerous companies have transitioned towards a data-driven approach, acknowledging the invaluable insights that robust data analysis can provide. This shift reflects a growing awareness of the strategic advantages and operational improvements that can be derived from leveraging data in decision-making processes.

The utilisation of data for decision-making processes necessitates a commitment to high-quality standards. It is crucial to ensure that data undergoes thorough cleaning to prevent the propagation of errors throughout the analysis and prediction phases. The integrity of decision outcomes relies on the cleanliness and accuracy of the underlying data. This emphasis on data quality safeguards the reliability and trustworthiness of insights derived from the data-driven decision-making process.

Data streams have witnessed widespread adoption in recent years, a trend attributed to their applicability in various fields. Prominent examples include the fields of IoT, Edge computing, and Industry 4.0, where the utilization of data streams has become increasingly prevalent. The unbounded nature and intrinsic velocity of data streams necessitate the development of purpose-built techniques to effectively manage and leverage their dynamic characteristics. The volume and velocity of data streams must also be addressed during data quality evaluation.

Data quality enhancement is achievable through data preparation, a process often guided by data quality assessment and profiling. By engaging in a comprehensive assessment of data quality, potential issues, and discrepancies can be identified, laying the groundwork for targeted data preparation efforts. Simultaneously, data profiling offers valuable insights into the inherent characteristics and patterns within the dataset, guiding the formulation of tailored data preparation strategies. This integrated approach makes it possible to systematically refine and elevate the quality of data.

This work aims to refine an already present framework for data preparation, designed exclusively for tabular data. The proposed additions and modifications are specifically

crafted to address the unique challenges posed by data streams. Unlike traditional tabular data, data streams necessitate a distinct approach, with careful consideration given to the dynamic factors of velocity and volume inherent in this type of data. The aim is to fortify the framework, ensuring its adaptability and effectiveness in accommodating data streams' continuous and high-speed nature.

The initial modification involves the realm of data profiling. A continuous profiling tool will be implemented to furnish real-time insights into the dynamic characteristics of the data stream. This tool is intended to offer instantaneous observations, allowing for a comprehensive understanding of the evolving nature of the data stream. The outcomes of the profiling will be used not only to gain real-time insights into the stream's characteristics but also to conduct a dynamic assessment of data quality across various dimensions, thereby providing real-time data quality metrics.

Another key enhancement encompasses refining the knowledge base utilized by the framework to recommend data preparation actions. A series of trials will be executed to thoroughly understand and evaluate which techniques prove to be the most suitable for the dynamic nature of data streams. This iterative exploration aims to enhance the knowledge base, ensuring that it incorporates the most effective and adaptive strategies for suggesting pertinent data preparation actions in the context of streaming data.

To some extent, the anticipated outcomes carry an element of unpredictability, which is intricately linked to the deliberate choice of the model. This interplay between the selected model and the projected results introduces a layer of complexity to the expected findings.

The primary goal of this thesis is to expand the existing architecture and address a gap in the literature that neglects the evaluation of the impact of outlier correction and missing values imputation on data stream analysis. Additionally, this work aims to provide a comprehensive data profiling tool specifically designed for data streams, addressing a notable absence in current literature.

Thesis Structure

Chapter 1 concisely presents the existing works that lay the foundation for this thesis and outlines the gaps in the literature it aims to fill.

Chapter 2 first presents the objectives of this study, followed by an exploration of the original architecture and how the study seeks to improve upon it. Additionally, there is an explanation of the practical experiments that have been conducted.

Chapter 3 provides a comprehensive list of the technologies and methods used in the actual implementation of the project. Additionally, this chapter includes a presentation of the datasets utilized during the evaluation phase.

Chapter 4 explains the evaluation methods employed and subsequently presents all the obtained outcomes, further illustrated in graphs and tables.

Chapter 5 discusses the findings and presents potential future work that can be undertaken in this field.

1 State of the art

1.1. Data Streams

A data stream is a countably infinite sequence of elements. Different models of data streams exist. Streams can be structured or unstructured. In the first case, elements follow a particular format or schema. These streams can be represented as a vector of elements, in which is possible to add, modify, and delete elements, like in relational databases. Another possibility is to represent structured streams with the cash register model, where only additions to the underlying vector are permitted. The last possible model is time series, where every element is treated as a new independent entry in a generally unbounded vector. On the other side, unstructured streams can have arbitrary contents, usually obtained by combining streams from different sources. [1]

Stream computing refers to the processing of a data stream. This processing must consider the large volume of data and the high velocity with which data is generated from different sources. The processing has to be done with low latency, ideally in real-time. This new paradigm is necessary for the scenarios that involve location services, mobile devices and sensors. It can be applied to IoT, sensors, market data, clickstreams, etc. [2].

[3] Data streams naturally carry a temporal dimension. The data streaming model discussed above assumes that the streaming computation starts at time t_0, and at any time t takes into account all the updates between t_0 and t. This is usually called "landmark model". In many applications, it is important to downgrade the importance of older items. Many time-decay models have been proposed for streaming data. One of the most prominent approaches is "sliding window". A sliding window over a stream is defined as a bag of the last N elements of the stream seen so far, for some nonnegative integer N. The size of the window can be fixed or variable and it can be tuple-based or time-based. [4]

Data streams can be managed using either a time-decay model, such as sliding window, or online algorithms. Sliding windows prioritise recent data subsets while online algorithms process data incrementally, adapting effectively to dynamic environments and constant changes. In both approaches, the goal is to extract meaningful insights from the data

stream while considering the challenges of processing data in real-time without running out of storage.

The substantial volume of data that data streams imply is a significant challenge to contend with. To face it, it is common to use a sampling procedure. In data stream sampling is required the application of techniques from traditional database sampling, and then significant innovations to handle queries over infinite-length streams. The main approach used for data stream sampling involves windowing. It is important to distinguish between stationary windows and sliding windows. In fact, sampling from a sliding window is much more complex. The difficulty arises because elements must be removed from the sample as they expire but the sample must be kept of a specified size. Many methods and approaches to do it are described in literature, like Bernoulli Sampling and Chain Sampling. [5]

Data streams cannot be managed simply using a RDBMS (Relational Database Management System) because streaming queries must allow order-based and time-based operations, require the use of approximate structures for aggregation and must react quickly to unusual data values. DSMS (Data Stream Management Systems) have been created for this purpose, and they use proper languages, like CQL or AURORA. [6]

1.2. Data Quality

Data has become increasingly important in recent years, and companies are now aware that data are valuable assets. Organizations that understand the value data can bring to their business have begun making data-driven decisions. To do this, it is necessary to analyze the data available. The amount of data needs to be as large as possible to improve the results. Companies gather data from all sorts of sources. These data, given their origin, may be of poor quality. The loss of quality can lead organizations to make poor decisions, resulting in large losses of money.

Data Quality (DQ) is an unavoidable issue in data analysis because it involves every single step of the pipeline. Low-quality data affects data management, machine learning models, data visualization, etc.

According to [7], every state of the real-world system should be mapped to at least one state of the information system. It is possible to have an incomplete representation of the real world in case some real-world entity is not covered in the information system. Other possible issues are given by ambiguous representations when more real-world objects are represented by the same object in the information system object. Finally, there is the

possibility of having meaningless states. These are states present in information systems, which do not exist in the real world.

Typical errors that influence DQ are missing values, outliers, duplicated data points, inconsistent format, and typos. For what concerns data streams, errors may include concept drift and historical changes. These errors may be due to a wrong or incomplete integration between two or more data sources. In the case of sensors' data, a fault in the device will bring errors.

In relational DB, DQ refers not only to the quality of the data itself but also comprehends the quality of the schema and its design. A particularly stressed aspect of data stream quality is time. In fact, the stream should not have delays and it is preferable that data follow a temporal order, especially when the timestamp is part of the stream.

DQ is often defined as "fitness for use". This means there is no strict definition, but it is relative and depends on the various uses requested for data. [8]. The variety of aspects affected by DQ makes it necessary to consider many points of view. In fact, it is required to assess DQ in many different ways. To do it, various measures have been developed and are called DQ Dimensions

1.3. Data Quality Dimensions

DQ is a multifaceted field and can be seen from different perspectives. This implies that it is not possible to measure the quality in a unique way, but it is necessary to use different methods, in order to consider every possible aspect.

Many different characteristics and definitions are used to calculate the quality of a dataset (or a data stream), and they are called Data Quality Dimensions (DQD). Each dimension measures a specific property and 179 dimensions were identified by [9]. According to them, the main dimensions can be divided into four different categories.

These categories are intrinsic DQ, contextual DQ, representational DQ and accessibility DQ. Intrinsic DQ captures the quality that data has on its own, contextual DQ considers the context where data are used, representational DQ captures aspects related to data representation, and finally accessibility DQ is related to accessibility and security of data.

Not every DQD can be estimated in a numeric way, some of these are only describable in a qualitative way. For example, it is difficult to quantify the representation or the accessibility dimensions.

1.3.1. Accuracy

There are many definitions for accuracy. [10] defines it as the closeness between a data value v and a data value v', where v' is considered as the correct representation of the real-life phenomenon that data value v aims to represent.

In [11], Klein says that the accuracy describes the systematic measurement error resulting from static errors in the measurement process, for example, due to miscalibration of the measuring method, or environmental influences on the measured values. The numeric value of this error is constant in sign and value. The accuracy of a numeric measurement value defines the maximal, absolute, systematic error a, such that the real value v' lies in the interval *[v-a; v+a]* around the measured value v.

ISO standard describes accuracy as "The degree to which the data has attributes that correctly represent the true value of the intended attribute of a concept or event in a specific context of use" [12].

To improve accuracy, the best way is to detect and delete inaccurate values. Be aware that inaccurate values can be meaningful, for example in the case of concept drift. The issue related to low accuracy is the unreliability of data.

1.3.2. Consistency

The consistency dimension captures the violation of semantic rules defined over data items. Integrity constraints are a representation of such semantic rules. [10] These constraints can be of two different types. The first type includes domain constraints, in which a data point has to be within a certain range or must follow some rules. Then, there are dependencies, which regulate the relationship between data. The simplest example is key dependency. Given a relational instance r defined over a set of attributes, we say that for a subset K of the attributes, a key dependency holds in r, if no two rows of r have the same K-values. Another dependency constraint, that can be used with data streams, is functional dependency. Given a relational instance r, let X and Y be two nonempty sets of attributes in r. r satisfies the functional dependency $X \rightarrow Y$, if for every pair of tuples t_1 and t_2 in r, holds : if $t_1.X = t_2.X$ then $t_1.Y = t_2.Y$

1.3.3. Completeness

There are different definitions for this dimension. For Batini and Scannapieco completeness represents the extent to which data are of sufficient breadth, depth and scope for the task at hand [10]. In relational databases, this can be characterized as the pres-

ence/absence and meaning of null values, assuming that the schema is complete. In [13], Geisler describes completeness as the percentage of missing values or elements to the number of values/elements collected.

For what concerns data streams, Klein states that completeness is the ratio of originally measured, noninterpolated values compared to the size of the analysed stream partition. Completeness can be related to a window and not only to the entire stream.[11]

1.3.4. Other Dimensions

There are some other significant dimensions, both for data streams and relational databases.

Data volume – The amount of raw data used to compute the result of a query. It is a very important parameter in the data stream domain where the volume of processed data is very large.

Timeliness - There are many definitions for timeliness. The most used definition in literature states that timeliness describes how current the data are for the task at hand [10]. Timeliness is closely connected to the notions of currency and volatility. Currency indicates the update frequency of data. Volatility measures how fast data becomes irrelevant. [14]

Confidence - [11]The confidence illustrates the statistical measurement error due to random environmental interferences. The confidence interval defines the bounds of the random distribution based on the variance of the data items. Confidence plays an important role during data stream processing, where selection or sampling is applied to reduce the data volume. During selection, an uncertainty range due to limited accuracy and confidence is generated around the threshold function, which may lead to falsely selected or unselected tuples, which distort the results of applied operators. [15][16] Confidence can be defined as the level to which users perceive the attributes of the data to be true and trustworthy in a given context of the use. Lack of confidence leads to unreliable reading.

Duplication - Existence of exactly the same records due to errors. This causes biases that will be problematic during data analysis and in the final decision step.[17]

Orderliness - For temporal streams, data are collected in chronological order. The data are considered out-of-order when they are recorded either ahead or behind schedule. Out-of-order data may influence the assessment of the degree of significance. [17]

1.4. Data Stream Profiling

Data profiling is the process of examining the data available from an existing information source, or in our case, from a data stream. The main purpose of data profiling is to collect statistics and information about the data and to create related metadata. The created metadata can be used to assess data quality.

The main issues in data stream profiling are given by the velocity of streaming data and the volume of big data. The first problem can be managed using incremental or continuous solutions, while the second can be resolved using one of the many existing sampling techniques. Data profiling methods are many and various. They can be divided into single-source and multiple-source profiling.

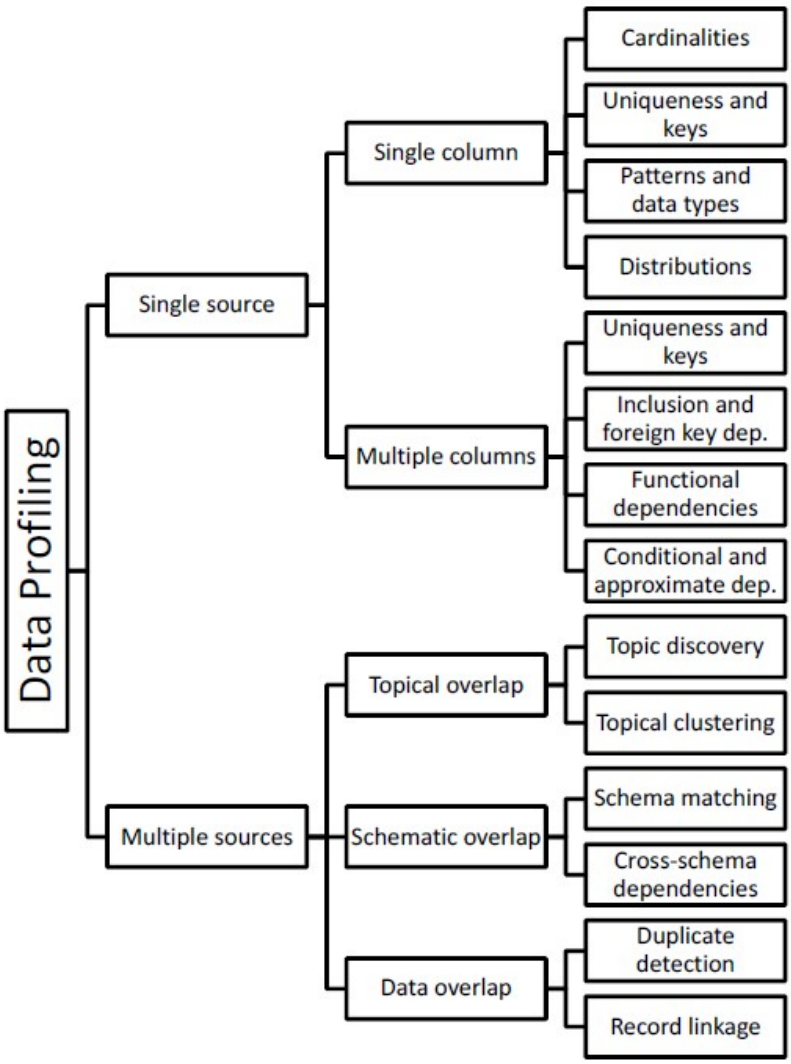


Figure 1.1: A classification of data profiling tasks.[18]

The most basic form of data profiling is the analysis of the single columns included in a source. The analysed statistics are mostly counts and distributions. The metadata traditionally computed are the number of values, the number of distinct items, the number of non-null values, etc. More complex techniques can create histograms and compute value distributions.

In single-source profiling, is included dependency detection. It is used to describe relationships among columns. The most frequent use case is to discover foreign keys, exploiting Functional Dependencies (FDs). The dependencies constraints can be relaxed using Conditional Functional Dependencies (CFDs). The multiple-sources profiling tasks are more complex and include topic discovery, schema matching and record linkage.[18] [19]. Multiple-source profiling is not explored in this thesis

In data stream applications, data arrive at high rates, thus profiling has to be performed continuously and efficiently in terms of both space and time.

Incremental profiling is used for datasets with periodic updates. It uses earlier profiling results to speed up the computation of new data. In simplest cases (as count, sum, etc.) metadata is computed associatively. In some other cases, as average, some intermediate metadata must be used. In more complicated cases, like median or clustering, a complete recalculation might be necessary.

Continuous profiling is used when it is necessary to update profiling results while data is created or updated. If profiling results can be expressed as a query, then continuous profiling can be achieved using DSMS (i.e. CQL queries) If this is not possible, continuous profiling methods need to be developed, and results can be shown in a dashboard.

1.4.1. Single Columns Analysis

The analysis of the values of individual columns is usually a straightforward task. The produced metadata in this step includes cardinalities, value distribution, data types and patterns and data completeness.

Cardinality is the number of elements in a set. In data profiling, some useful cardinalities can be the number of rows in the table, the number of distinct values, etc. In a streaming data scenario, they must be updated continuously. For some statistics, like the number of rows in a table it is quite simple, since it can be managed through a simple counter. Another value that can be computed incrementally is the number of null values in a data stream.

A more difficult task about cardinalities is to count the number of distinct items in a data

stream. There are many particular algorithms to do it, and they are mostly based on sampling and hashing. An example can be Alon, Matias and Szegedy's algorithm (AMS). AMS algorithm is based on linear hash functions. The estimation of distinct items is based on the trailing zeros of the hashed values. The error guaranteed in the AMS algorithm is: for every r>2, the ratio error of the estimation is at most r with probability at least 1-2/r. [20][21]

Another common task of single-column analysis is the study of value distributions. The simplest values to be computed are minimum and maximum. To do it is enough to check if the arrived value is smaller than the minimum or larger than the maximum. The average value can be computed using the sum and the count previously computed in cardinalities. A more complex task is the research of the most k frequent item in a data stream. This can be achieved through the count-min sketch. This algorithm tracks the approximate count of items for each value over the data stream using a 2-dimensional array as data structure. [22]. There exists some specific algorithm to compute top-k frequent items over data stream.[23]

A further key point of value distribution is quantiles. There are many algorithms to compute approximate quantiles over data stream. One of the most important is the one introduced By Greenwald and Khanna in 2001. The proposed data structure is an ordered sequence of tuples which corresponds to a subset of the data points seen so far. Every tuple contains the value and information about its rank. [24]. More methods to compute quantiles in an approximated way are investigated by Chen and Zhang [25]

1.4.2. Dependencies

Dependencies are metadata that describes relationships among columns. A common goal of data profiling is to identify suitable keys for a given table, both primary keys and foreign keys. Another form of dependency that is also relevant for data quality is functional dependency (FD). An FD states that values in one set of columns functionally determine the value of another column. This constraint can be relaxed using partial dependencies, which hold only for a subset of records, and conditional dependencies, which hold only for specified conditions.

In this thesis, the method used to capture functional dependencies is a window-based version of Apriori algorithm [26]. However, other algorithms to detect FDs in data streams are already in the literature. A notable example is DynFD, in which a dynamic lattice is updated and pruned according to insertion and deletion in the dynamic dataset.[27]

1.5. Data Preparation actions

As discussed in Section 1.2, Data Quality is a crucial concern for companies and organizations. It is necessary to have well-structured data, as well as complete data. Another key stage to consider is data cleansing in which noisy and inconsistent data points must be addressed. This is necessary not only for simple analysis but for the correct use of the machine learning models. Most of the time, ML models require complete data and achieve a better performance when data is clean. Training an ML algorithm with a dataset which contains outliers may lead to a bias in prediction or to overfitting. The ensemble of processes aimed at rectifying this is called data preparation.

Data preparation includes all the activities meant to improve the quality of the dataset before the analysis. These actions comprise data exploration, structuring, and cleaning and are performed in the early stages of a data processing pipeline. Data scientists spend approximately 80% of their time preparing data and about 20% on actual model implementation and deployment.[28]

Table 1.1 shows some of the data preparation actions frequently used on data streams. The activities that have been used in this thesis are most of these and are explained in this Section.

Data Inspection is the process of analyzing and collecting data from different sources. The main step is to isolate all the notable values. Once the data is extracted from the raw stream it is possible to locate problematic data points as missing values, outliers or duplicates. Once that data is accurately inspected it is possible to structure it in a predefined schema. This is called **Data Structuring** and includes tasks for the creation, representation and structuring of information.

Data Validation includes constraints and rules to check data correctness, consistency and other data quality dimensions.

Data Filtering generates a subset of the whole dataset removing irregular data values. In a data stream context, filtering can be time-related as it happens in windowing. Windowing is required because of the unbounded nature of data streams, and stateful analysis must be performed over portions of data. **Data Cleaning** refers to removal, addition, or replacement of inaccurate data values with other values which can be inferred from the portion of the data stream which is reputed clean.

In a Big Data context, it is common to use **Data Reduction** techniques. The goal is to obtain an accurate and adaptable model that at the same time has a low computational

complexity in order to respond quickly or even in real-time. There are several ways to do data reduction, and they act both on features and data instances [29]. In contrast to this, it may be necessary to have additional information, for example, to train a deep learning model, it is required to have a great amount of data. **Data Enrichment** manages to do it and involves augmenting existing data with new or derived data, possibly even from separate sources. [28]

In data streams, it is important the concept of time and its management. Some analysis requires the assessment of the temporal order in the dataset. This constraint must be respected and checked during data ingestion or data fusion. Event-time disorder can be observed due to two main reasons. The former is the communication medium used to forward the data, for example, UDP protocol. The latter is due to join from multiple streams. According to [30], there is a dedicated operator to merge data streams enforcing event-time ordering.

Another problem related to the concept of time is the delay between event time and processing time. This can be an even bigger problem when windowing is used. It may happen that a data point is processed when its window has already expired and this can bring an inconsistency in stream analysis. To overcome this issue, some DSMS, like Apache Spark, have a feature called Watermarking. The watermark is a threshold which specifies how long the system waits for late events. If an event is processed within the watermark, it will update a windowed query. Otherwise, it will be dropped and not processed. [31]

Activity	Method
Data Inspection	Extracting fields Locate missing values Outlier detection Duplicate Detection
Data Structuring	Insert values in a defined schema
Data Validation	Check if values are in a determined range Check if values satisfy integrity rules Check Dependencies consistency Check Datatype consistency
Data Fusion [30] [32]	Sort according timestamp Machine Learning based methods Probabilistic methods Statistical methods Knowledge-based methods
Data Aggregation	Summarize data Feature extraction
Data Filtering	Noise filtering Missing values filtering Windowing
Data Cleaning	Missing values correction Outliers correction Data format standardization Inconsistent data deletion Duplicates deletion
Data Reduction [29]	Feature selection Instance selection Sampling
Discretization [29]	Incremental Discretization Algorithm Online ChiMerge Algorithm
Temporal Operations	Watermark Temporal order assessment

Table 1.1: Data Preparation Actions

1.6. Literature Gaps

During the initial research phase, certain gaps were identified in the existing literature. The primary gap pertains to data stream profiling, where although techniques for functional dependencies have been developed, continuous profiling for data distribution is notably absent. While Naumann discussed continuous profiling in 2013 [18], this aspect has not been significantly addressed in subsequent literature. Furthermore, beyond the research on functional dependencies, there is a lack of tools or frameworks specifically dedicated to data stream profiling.

Another gap addressed by our work is associated with the evaluation of the impact of data preparation actions on data stream analysis. There exist comparisons between outlier detection methods that focus on recognition capabilities, and there are many assessments of the influence of data preparation on tabular data analysis, but there is not any significant result for what concerns data streams. Our study aims to bridge this gap by systematically assessing and comparing the effectiveness of various data preparation actions in the context of their impact on machine learning model performance

In conclusion, with regard to data quality, the literature has described DQ dimensions specific to data streams, particularly in the context of sensor networks. While frameworks for data quality assessment exist, a notable gap remains in the absence of a ready-to-use tool that allows the representation of data quality on a dashboard. Our research aims to address this gap by exploring and potentially developing a tool that facilitates the representation of data quality metrics in a user-friendly dashboard format.

2 | Methodology

Data quality is crucial in many use cases, especially when the analysis involves predictions and data-driven decisions. The completion of data preprocessing and preparation is a protracted task, which requires specific criteria to be followed in a particular order. Numerous steps are required to be carried out to ensure optimal results.

Section 2.1 illustrates the goals of this thesis. Section 2.2 explains the present architecture and how this study tries to enhance it. Section 2.3 presents the methodology followed in the experimental part.

2.1. Method Objectives

The primary objective of this study is to comprehend the behaviour of data streams during the phases of data profiling and data preparation, and how some of the techniques designed for tabular datasets can be adjusted for the streaming scenario.

This thesis aims to improve an existing framework outlined in Section 2.2. The enhancement comprises three tasks. The first completed task involves the development of a continuous profiling tool, which is built by adapting techniques designed for tabular datasets. Further details can be found in Chapter 3.5. A streaming adaptation of data quality assessment is performed alongside with data profiling.

The second aim of this work is to conduct a comprehensive comparison of data preparation actions and their impact on the performance of machine learning models. Various data preparation methods are evaluated under different conditions of data quality. This comparative analysis seeks the most suitable approach for handling data streams.

An additional stage involves experimentally observing the same data preparation actions using a tabular dataset. This allows a comparison between the behaviour of data streams and tabular datasets. The objective is to understand the impact of identical data preparation procedures on the analysis of stream and tabular data.

The details of these two aspects of the work are discussed in Section 2.2, while the exper-

imental results and their evaluation are presented in Chapter 4.

2.2. Architecture

The existing architecture is designed to provide an environment for data preparation. The process workflow is illustrated in Figure 2.1 and is meant to be used by the end user to receive suggestions for the best data preparation actions to execute for their particular dataset.

The recommendations are created after data exploration, profiling and quality assessment. The outcomes of these phases are used to suggest the most suitable data preparation actions, chosen from a knowledge base that contains all the required information to support the execution of this process.

This study aims to extend the application of the existing pipeline beyond tabular datasets to incorporate data streams. Such an expansion requires a distinct approach, one that is responsive to the dynamic and evolving nature inherent in data streams. Two distinct approaches were adopted to address this challenge based on the requirements. Online processing was applied for data profiling, whereas sliding windows were utilised for data quality assessment and preparation.

In data stream management, sliding windows maintain a constant subset of the most recent data points. As new data arrives, the window slides forward, incorporating the latest elements and removing the oldest ones

The data profiling stage is conducted continuously, with its findings being updated after every newly ingested data point. This necessitates an incremental approach, providing the capability to obtain real-time updates on evolving data characteristics.

Beyond data profiling, data quality assessment techniques have been adapted to a streaming scenario. Utilizing a windowed approach to compute data quality dimensions, obtaining an overview of data quality at each point in the stream becomes feasible. This approach enables a comprehensive understanding of the trend, whether it involves a decay or improvement in data quality over time.

The latest enhancement to the established architecture focuses on the knowledge base. This study evaluates and compares various data preparation actions to determine their efficacy for data streams. The identified optimal data preparation actions will be suggested to the end user, particularly in scenarios where the input data source is a data stream.

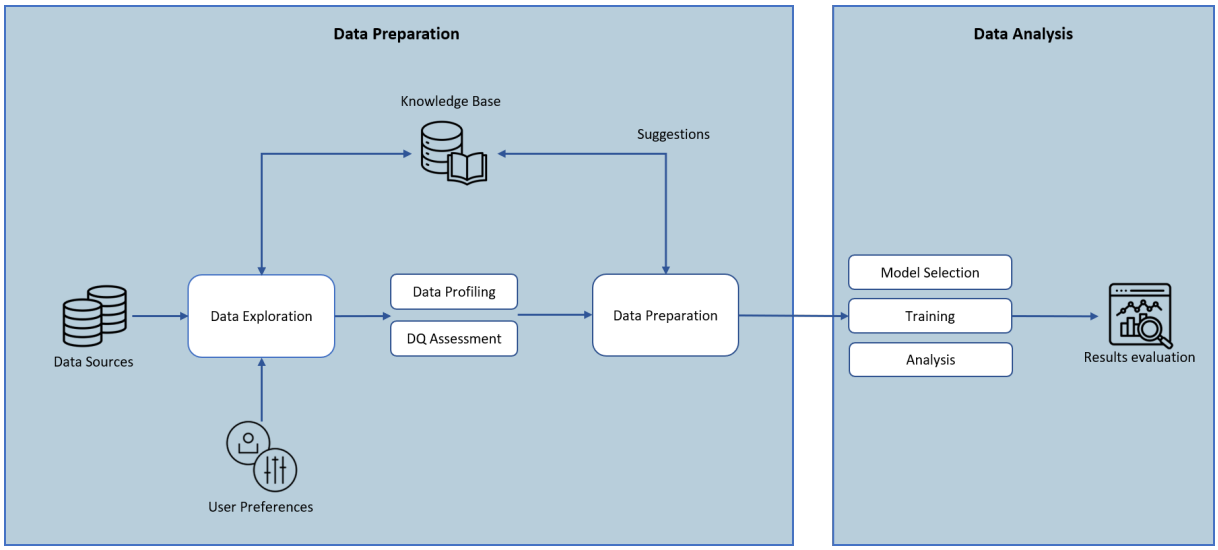


Figure 2.1: Architecture pipeline

The process in the presented pipeline initiates with an input phase wherein the user u selects the dataset d and specifies preferences up. The preferences up include information about the chosen dataset and the analysis to be executed.

In the existing architecture tailored for tabular datasets, data exploration is conducted to provide users with an initial understanding of data characteristics. Subsequently, the system performs data profiling — a detailed analysis aimed at extracting comprehensive information about the dataset. This data profiling step serves as a foundational process for identifying and addressing any critical issues inherent to the data.

The dataset d is initially read, and from this dataset, a comprehensive profiling report is generated by leveraging various data profiling features, denoted as dpf. These features collectively offer a detailed analysis of the dataset, providing insights into its characteristics. Subsequently, data quality is assessed, and the metrics associated to data quality dimensions dqd are computed through the use of the data profiling report, with each dimension focusing on a specific aspect of Data Quality. By assessing the most critical dqd, the system identifies potential challenges and areas for improvement in Data Quality (DQ).

Based on the analysis of dqd, a suitable data preparation action dpa is retrieved from the knowledge base. The knowledge base contains a repository of predefined data preparation actions tailored to address specific DQ issues. The suggested data preparation action is then presented to the user u, offering guidance on enhancing the dataset's quality before proceeding with further analysis or modelling.

Following the preparation phase, the data analysis stage is conducted. The model selected by the user u in the preferences up is trained, and the analysis is executed, enabling u to evaluate the results.

This work does not entail a complete overhaul of the existing framework but adapts certain phases of the process to accommodate data streams.

The modifications made to the architecture include data profiling being computed in a continuous way and data quality assessment being executed with a windowed approach. The dataset d is now a data stream, and it is read row by row, and each data point contributes to the update of the data profiling feature dpf. At the end of each window, dpf is utilized to compute the data quality dimension dqd. Given that d is read one point at a time, the real-time update of dpf and dqd enables a continuous evaluation of DQ.

In this research context, data preparation and analysis stages are structured around a windowed approach. This method strategically focuses on the most recent data points within a defined window, promoting an ongoing refinement of the model as it adapts to the evolving nature of the data stream. By prioritizing recent information, this approach seeks to enhance the model's responsiveness and effectiveness in capturing the dynamic patterns inherent in data streams.

A fixed-size window w is established, initialized with the initial x data points from the dataset d. Once the window reaches its capacity, the data preparation action dpa is triggered, impacting the samples within the window. The selection of dpa is guided by the knowledge base, taking into account data quality dimensions dqd and user preferences up. The analysis is then performed on the window w, yielding predictions on the target variable computed by an ML model. The resulting predictions are stored for subsequent evaluation. Following the completion of data preparation and analysis, the earliest y samples in w are discarded, making space for an additional set of y data points. This sequential process continues as the data stream progresses through the pipeline until the conclusion of the data stream, allowing for the analysis and evaluation of the obtained results.

2.3. Knowledge Base Augmentation

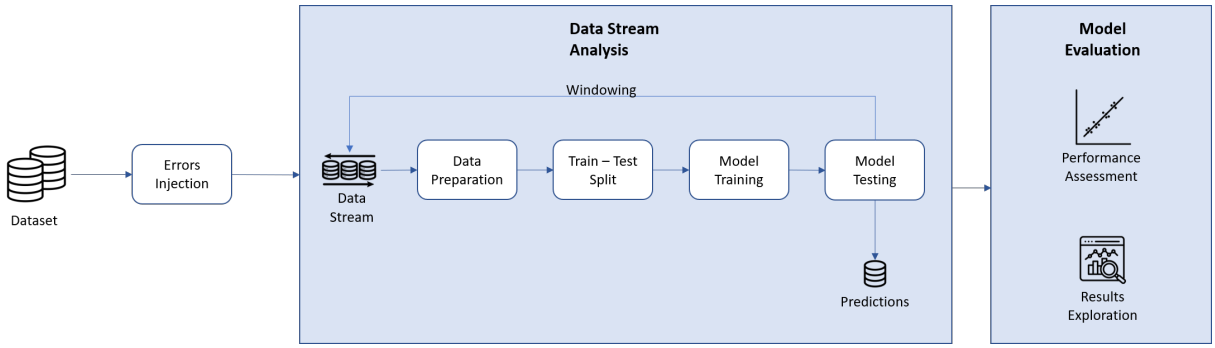


Figure 2.2: Experimental pipeline

This methodology seeks to compare diverse data preparation actions aimed at enhancing the quality of machine learning analyses. The study comprises multiple steps based on different assumptions. The work outlined in this section aims to enrich the existing knowledge base through experimental research. It is initially presented in a formal manner, while Section 2.3.2 provides a practical explanation of all the designed experiments.

The primary objective of these experiments is to enrich the knowledge base with insights and information specifically related to data stream preparation actions.

The knowledge base contains the required information to support the execution of the pipeline illustrated in Figure 2.1. Its content includes information regarding data preparation processes, data quality dimensions, machine learning algorithms, and their interconnections. The additions made after this study contribute to the knowledge base by integrating information related to data streams based on an experimental evaluation.

This experimental evaluation follows the methodology depicted in Figure 2.2 and outlined below is utilized to understand the influence of data preparation on data stream analysis. For completeness, the methods have been assessed across diverse datasets with varying levels of data quality.

Each dataset, denoted as d, is closely related to a specific machine learning model and, consequently, to its performance metric, denoted as m. The datasets undergo testing across various data quality levels, made possible by the deliberate injection of errors into the original dataset.

The original dataset d is intentionally contaminated with errors. Errors concern a specific data quality dimension dqd. Multiple datasets are generated during this phase, labelled

as $d_1, d_2, ...$ Each of these datasets d_i contains different errors, in order to test data preparation under different quality conditions. Throughout the injection process for each dataset d_i, a corresponding mask mx_i is created, representing all the rows in the dataset where an anomaly was injected. The subsequent approach is systematically executed for each of these datasets d_i.

For each dataset d_i, data points are ingested one at a time. These data samples are collected within a window w of size x. The data preparation action dpa_j is applied to the window w, and subsequently, the window w is split into train and test sets. The train set is used to fit the machine learning model, while the test set is employed to yield the predictions. Once this phase is over, y data points are dropped from w, creating room for additional samples.

The predicted values and the actual values present in d_i are saved after each sliding window w analysis. Once the entire stream is processed and each window w has been analyzed, metric m_{ij} is computed to assess the machine learning performance. All the computed metrics m_{ij} are then evaluated to comprehensively compare all data preparation actions dpa_j. This evaluation aims to determine which data preparation action is the most suitable.

An additional investigation is conducted following the same procedure described earlier. Two distinct data preparation actions denoted as dpa_j and dpa_k, are executed in this case. The windowed approach remains consistent with the aforementioned process, but in this instance, we obtain two metrics, m_{ijk} and m_{ikj}. These metrics allow us to comprehend which combinations of data preparation actions yield greater efficacy. They also enable an understanding of whether variations in the sequence lead to noticeable differences in the final outcome.

This review aims to assess the effectiveness of outlier detection methods. The pipeline used in this experiment differs slightly from the one shown in Figure 2.2. Here, no machine learning tasks are involved in data preparation, which instead revolves around detecting and subsequently evaluating outliers.

An outlier detection method od_j is performed on dataset d_i, in a windowed manner, as discussed previously. w is no longer analysed with the help of a machine learning model. Instead, the detected outliers o_{ij} are compared with mx_i, yielding a metric m_{ij}. This metric can be used to comprehend the performances achieved by each od_j.

One of the goals of this work is to compare some methods of outlier detection and data imputation and evaluate their impact on machine learning tasks. The ML algorithms and

the metrics used to assess them are illustrated in Chapter 3

From this point onward, we delve into the implementation of the outlined pipeline, providing a step-by-step exploration of how the theoretical concepts and strategies are translated into practical experiments and applications.

2.3.1. Errors Injection

The initial phase of this experimental methodology involves the deliberate introduction of errors into the dataset. This procedure systematically assesses the efficacy of various data preparation techniques. The intentional injection of errors is intended to degrade two distinct dimensions of data quality: completeness and accuracy. Completeness is compromised through the introduction of missing values, whereas the introduction of outliers undermines accuracy. This controlled manipulation of the dataset allows for an evaluation of the performance of different data preparation approaches under varied conditions.

The introduction of errors not only generates an alternative dataset but also results in the creation of multiple datasets characterized by varying levels of data quality. This variation in data quality levels is designed to yield a more comprehensive set of results, facilitating an understanding of the trends in machine learning performance across diverse data quality scenarios.

Three different data manipulators are created. The first is designed to insert missing values to impair completeness. The second injector undermines accuracy by introducing outliers. The last is created as a combination of both these strategies. This stage aims to recreate a dataset that mirrors real-world conditions in which errors can occur.

The algorithm employed for this task adopts a probabilistic approach. It is initiated by reading the entire dataset. Subsequently, it determines which values will be replaced by either a null value or an outlier, generating a mask. The computation of this mask is done randomly, aligning with the specified level of quality. All values identified by the mask are then replaced with the defined errors.

In the first two cases, each p_i in $[0.1, 0.2, 0.3, 0.4, 0.5]$ can be marked as an outlier row with p_i probability. Each row could have a variable number of true values according to predefined probabilities. The masks $m_1, m_2, ..., m_5$ are created in this way and are used to create five different datasets $d_1, d_2, ..., d_5$ in which the true values in the mask m_i correspond to an outlier or a missing value in the dataset d_1

In the final scenario, where both outliers and missing values are introduced into the

dataset, the rationale remains consistent. However, there may be missing values or outliers for each flagged row, but not both. This assignment is made randomly with an equal probability, ensuring a comparable number of rows containing outliers and rows with missing values.

2.3.2. Experiments

Six different experiments have been designed to correctly assess the impact of data preparation on the prediction. The investigated data preparation actions are data imputation and outlier detection. The trials are illustrated in Table 2.1 and discussed below. All the presented experiments follow the generic pipeline, which is presented in Section 2.3.

Experiments concluded

N	Injection	Task
1	Missing Values	Data Imputation
2	Outliers	Detecting Outliers
3	Outliers	Outlier standardization
4	Outliers	Outlier correction
5	Outliers + Missing Values	Outlier Correction and Data imputation
6	Outliers + Missing Values	Data imputation and Outlier Correction

Table 2.1: Data imputation methods comparison.

The initial experiment aims to determine which data imputation technique assists in building a superior machine learning model. Only missing values are introduced in the clean data set, resulting in five datasets with varying degrees of completeness. Following the execution of a data imputation technique on the dataset, a machine learning task is then executed. This procedure is applied to each of the methods represented in Section 3.7.

The next three trials involve the introduction of outliers. Similar to the previous instance, five datasets are generated. However, the data quality aspect under consideration here is accuracy. The first iteration assesses the effectiveness of each outlier detection method explained in Section 3.6. The evaluation involves comparing the detected outliers with the injection mask outlined in the previous section, with the F1-score serving as the metric for goodness assessment.

The following aspect to consider is the influence of outlier detection on machine learning outcomes. In fact, experiments 3 and 4 aim to comprehend which outlier detection method is the most suitable for regression and classification.

The third experiment conducts outlier detection on the injected dataset, followed by the normalization of all detected anomalies to a standard value. Thus, the machine learning prediction is executed after the standardization.

The unique distinction between the third and the fourth trial lies in the correction method. Specifically, the only assessed correction in the third test involves standardizing outliers. Conversely, in the fourth test, all the methods elucidated in Section 3.7 are employed to rectify the outliers. The objective of the fourth trial is to identify the most effective combination of outlier detection and correction techniques.

The two last experiments are based on outlier and missing values injection. Their aim is to assess potential correlations between the sequence of data preparation actions and the efficacy of machine learning predictions. Specifically, the experiments are designed to conduct outlier correction followed by missing values imputation, and vice versa.

All these experiments are also conducted on a tabular dataset. This is performed to assess the variations in the impact of data preparation on different types of data. The methods applied to the tabular dataset are batch-based, in contrast to the windowing approach employed for data streams.

3 | Experimental setup

This chapter is dedicated to explaining the technologies and methods employed in both the development of the pipeline and the implementation of the experiments. Section 3.1 provides a comprehensive overview of all the technologies, mostly Python libraries, used in the process. Section 3.2 explores the datasets employed for conducting the experiments. Section 3.3 outlines the machine learning methods employed for prediction, while Section 3.4 gives insights into the metrics used for evaluating these predictions. Section 3.5 details the approaches to data profiling and assessing data quality. Finally, Sections 3.6 and 3.7 elaborate on the data preparation actions tested in the experiments discussed in the previous chapter.

3.1. Technologies

This section enumerates the technologies employed in the implementation of the previously described pipeline. The majority of these technologies are Python libraries, given that the software has been developed using this programming language.

The platform utilized for simulating a data stream is Apache Kafka, an open-source stream processing platform. The Kafka server is executed within a Docker container, accompanied by a Zookeeper server, both established on the same machine where the code is executed. The interaction between Kafka and the Python code is facilitated through the 'kafka-python' package, available in the PyPI repository.

All the subsequent packages are sourced from PyPI or open GitHub repositories

- **NumPy** - Package for scientific computation. It provides multidimensional array structured objects and enables fast operations on these arrays and matrices.
- **Pandas** - Package for data analysis and manipulation. It contains data structures for tabular data and time series. It is primarily used for DataFrames, data structures that facilitate easier and faster data manipulation.
- **scikit-learn** - Package for machine learning analysis. It contains numerous machine

learning models and various tools that support them.

- **PyOD** - Package for outlier detection. It encompasses various methods designed to detect outlying data points in multivariate datasets.
- **River** - Package for streaming machine learning. It adapts numerous scikit-learn methods to streaming data and implements specific techniques tailored for streaming scenarios.
- **Matplotlib** - Package for plotting and visualization creation. It enables the creation of graphs and figures, including interactive visualizations

Other used packages include **'datetime'** for transforming and standardizing time formats, **'csv'** and **'pickle'** for opening, reading, and writing files.

3.2. Datasets

Three distinct datasets are employed in this thesis. Two of these datasets simulate a dynamic data stream, capturing the evolving nature of data over time. In contrast, the third dataset adopts the traditional tabular format. This diverse selection enables a comprehensive exploration of the impact of various data preparation actions on streaming data, facilitating a meaningful comparison with a tabular dataset. A summary of the data characteristics can be found in Table 3.1.

The first dataset used is the Beijing Multi-Site Air-Quality Data Set, hereafter referred to as the **AirQuality** dataset[33]. This dataset encompasses hourly air quality measurements from several monitoring sites. In pursuit of the cleanest data possible for our work, a careful selection of the most reliable observation points was undertaken. Subsequently, the dataset underwent a cleaning process, where outliers were removed, and missing values were imputed. This meticulous preprocessing aimed to ensure a pristine dataset for subsequent error injection experiments. The target value is 'PM2.5', and the executed machine learning task is regression.

The second dataset is the **NEWeather** dataset, where daily weather measurements are recorded [34]. This dataset comprises data from an Air Force Base in Nebraska and is sourced from a National Oceanic and Atmospheric Administration database with records from over 7000 stations worldwide. Notably, this dataset is already pre-cleaned, without any missing values, making it readily available for use. The classification task is performed on a binary target variable representing the presence of rain on a given day.

The final dataset is a tabular dataset used exclusively for the last evaluation, in which

the impact of data preparation on data stream analysis is compared to the impact of data preparation on tabular data analysis. In this work, this dataset is referred to as 'Electrical' [35]. It simulates the dynamics of an electrical grid, featuring two target values associated with the stability of the electrical grid. The first target is represented by real numbers, enabling regression testing, while the second target is a label, facilitating classification testing. This dual-target structure allows for a comprehensive evaluation of both regression and classification tasks.

Dataset Characteristics

Datasets	Instances	Features	Attributes	Associated Task
AirQuality	70128	15	Numerical / Categorical	Regression
NEWeather	18159	9	Numerical	Classification
Electrical	10000	14	Numerical / Categorical	Reg / Clf

Table 3.1: Employed Datasets

3.3. Machine Learning algorithms

The machine learning algorithm utilized in this thesis is **Random Forest** (RF). It is an ensemble learning method that combines the output of multiple decision trees to make predictions. This technique is capable of handling both regression and classification problems, accommodating numerical and categorical attributes. In this work, the scikit-learn implementations RandomForestRegressor and RandomForestClassifier are employed.

In hindsight, the Random Forest algorithm was not optimal for our work, as it demonstrated robustness to outliers and required minimal data preparation. An additional test was conducted with K-Nearest Neighbors (KNN), a distance-based technique that, like Random Forest, can be employed for both classification and regression tasks. Unlike RF, KNN is less resistant to outliers, making it a potentially better choice for the initial stages of the analysis.

Before applying the machine learning model, the dataset - or in this case the single window - is divided into training and testing sets. This division is performed using a scikit-learn function, where 67% of the window is assigned to training and the remaining 33% to testing. This approach allows for an accurate analysis, where past samples are used to predict future ones.

3.4. Performance Metrics

This section presents an overview of the metrics employed to evaluate the performance. This encompasses the metrics utilized for assessing the performance of machine learning tasks. Additionally, the metrics employed to evaluate data quality are detailed in Section 3.5.

To evaluate the regression model performance, the **R2-Score** is employed. Widely used in statistics, the R2-Score serves as a common measure to assess how effectively a regression model predicts a target variable.

The formula used to compute R2-Score is Equation 3.1, in which n is the number of observations, y_i denotes the observed value, $\hat{y}_i$ represents the predicted value and $\bar{y}$ is the mean of the observed values

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.1}$$

The classification model is assessed using the **F1-Score**, a measure of accuracy in binary classification. This metric is derived from precision and recall. Precision represents the ratio of True Positives (TP) to all positives, indicating the model's ability to predict correct positives. Conversely, recall is the ratio of TP to the sum of TP and False Negatives (FN), providing a measure of the model's capability to identify relevant data.

The F1-Score is indeed the harmonic mean of precision and recall. It is commonly used in classification tasks, especially when there is an uneven class distribution. The F1-Score provides a balanced measure by considering both precision and recall, making it a suitable metric for scenarios where both false positives and false negatives are crucial considerations in model evaluation. It is computed as

$$F1 - Score = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.2}$$

The F1-Score has been utilized as a metric to evaluate the effectiveness and performance of outlier detection methods, providing a comprehensive assessment of their ability to accurately identify anomalies within a given dataset.

3.5. Stream Profiling

Data quality assessment and improvement need at first a proper inspection of the given data. Data profiling is a good way to gain insight into the characteristics of the dataset. As reported in 1.4, data profiling comprises many different explorations and they generally are different between single source and multiple source data. In our case, we have a single source, so we focus on single and multiple column analysis.

The simulation of the stream takes place using Apache Kafka. A Kafka producer is established and communicates with a Kafka broker. The producer reads the entire dataset from a .csv file and transmits it to the broker row by row. The producer has only added the data transmission time in the original row; no other processing is carried on the data.

Subsequently, a Kafka consumer reads each message on the broker and operates on the data to do either stream profiling or data preparation. For the sake of simplicity, the Kafka server, the producer, and the consumer are run on the same machine.

The consumer needs to receive from the user some parameters concerning the data stream's structure. The most important parameters are the names of the features and the data types that the producer will receive in each single message. The date-time format of the stream is another important parameter to set. This is necessary to correctly handle both the timestamp generated by our Kafka producer and the timestamp present on each dataset. The last optional parameter is how the missing value is represented in the original dataset. It may occur that a null value is marked with a specific value, so it would be valuable to clarify the placeholder used in order to identify and process it accurately.

When data is read from the consumer, the first stage is a preliminary processing in which each value is divided from other values, and missing values are marked. This function receives in input a string, which was directly read from the broker, and returns a list in which the data point is clearly readable and is fit for use.

The described methodology is used repeatedly throughout this work and represents the starting point for both stream profiling and data preparation tasks, explained respectively in this chapter and in Section 2.3.2

3.5.1. Data Distribution

One of the primary objectives of data profiling is to comprehend how data is distributed. In a static scenario, this can be easily achieved by analysing all the present data, but when a data stream is involved, it may be necessary to perform computations in a continuous

way.

The minimum and maximum values for each dimension in our dataset are the simplest values to collect. It is sufficient to keep track of these values by comparing each received value with the actual minimum and maximum. By doing so, the extremes of each feature can be reported at every point of the stream.

Two other characteristics of the dataset that can be easily computed in an incremental way are mean and standard deviation. Equation 3.3 shows how to compute mean in an incremental way, and Equation 3.4 shows how to compute the standard deviation. In these equations, $count_t$ represents the total number of values received at time t

$$\overline{x_t} = \frac{\overline{x_{t-1}} \times count_{t-1} + x_t}{count_t} \tag{3.3}$$

$$std_t = \frac{count_{t-2}}{count_{t-1}} \times std_{t-1} + \frac{1}{count_{t-1}} \times (x_t - \overline{x_{t-1}})^2 \tag{3.4}$$

The topic of space occupancy is particularly important due to the vast volume of data streams. The next two accomplished tasks are critical in this respect. To tackle it, they use approximations and specific data structures.

Quantiles are fundamental in accurately describing the distribution of data. The algorithm employed for its incremental computation was developed by Karnin, Lang, and Liberty in 2016. [36] Their algorithm, called KLL, is a randomized approach that approximates the rank of any query with a negligible error. In our software, a KLL instance is constructed for each expected feature during the system initialization. Then, during execution, for each processed data point, the quantile structures are updated in accordance with each row.

The final step regarding data distribution is the computation of the top-k frequent items within the data stream. The chosen implementation includes a count-min sketch, which is a particular data structure based on hashing. This architecture allows us to find approximately the heavy hitters in a data stream, with a logarithmic space complexity. [37]

3.5.2. Data Quality Assessment

This section explains how some data quality dimensions are assessed in our software. The DQ dimensions present in this part are timeliness, completeness, consistency, and partly accuracy.

As stated in chapter 1, timeliness is frequently defined as the dimension that describes how current the data are for the task at hand. In this work, it is calculated by dividing the number of values received on time by the total number of values. A data point is labelled as delayed if it is handled after a predetermined time interval. The value of this threshold is adjustable by the user. Similarly, completeness is calculated by the proportion of non-null values to total values. These two DQ dimensions are basically reported as a number between 0 and 1 or, for the sake of clarity, as a percentage.

The notion of consistency used in this work is related to the concept of functional dependency. During the data profiling process, functional dependencies are computed within each window. The technique used to detect FDs in data streams is borrowed from association rule mining and it is the Apriori algorithm. It was selected because it allows the choice of minimum support and confidence, and given that the tested datasets are almost only numerical, it was difficult to find exact dependencies. The chosen implementation is taken from 'efficient-apriori' library. It is executed once in a window, and the detected dependencies are compared with the ones detected previously. If some rule is not valid anymore it is truncated from the list of actual FDs in the stream. Every violation of the rules is marked and counted. Consistency is computed by dividing the number of these violations by the number of total checks done in FDs.

Accuracy is determined by excluding invalid, received values and assessing the validity of data points using the interquartile range (IQR). The IQR is the range between the first and third quartiles of a dataset, which represent the 25th and 75th percentiles, respectively. Any values that fall outside this range are identified as anomalies. In this study, the IQR is computed incrementally, sample by sample, using the quantiles from the KLL data structure, which was discussed in the previous section of this work. The accuracy is ultimately evaluated by dividing the count of valid entries, denoted by those within the IQR, by the total count of values. The interquartile range (IQR) method may be replaced with more advanced techniques for detecting outliers and anomalies tailored for data streams.

All of the reported data quality dimensions are calculated per window, allowing the user to monitor the decay or the improvement of the overall data quality in the stream

3.6. Outlier Detection Methods

Outliers are data points that differ significantly from other data points. They can be caused by faults in the system, like in a sensor scenario, in which a breakdown can occur and sensors may produce misleading data. In a data stream, in which data continue to

arrive, these data points may be more significant than they appear. In fact, it is possible to have a particular trend over time, and this evolution may not be captured from the model, which will continue to mark these points as outliers. This problem is called concept drift.

The first tested method is **Z-Score**. It is a statistical-based approach. It uses mean and standard deviation to classify a data point as an outlier. This score is computed as $Z = \frac{x-\mu}{\sigma}$, in which x is the analysed value, μ is the mean of all the values and σ is the standard deviation. Since our data is multivariate, this is computed for each feature of the row, and since it is a data stream, Z-Score is computed in an incremental way. The mean and the standard deviation are calculated value by value. The data point is flagged as an outlier if at least one of his values has a Z-score greater than a certain threshold. The fixed threshold in this thesis is 3.

The second approach is distance-based and it **Local Outlier Factor** (LOF). This is a density-based approach and uses the concept of local density, which is computed on the distance of the k nearest neighbours. Comparing the local density of an object with the local density of its neighbours, the algorithm detects the outliers, which are the points which have a lower density with respect to their neighbours. Some incremental version of this algorithm has been developed and described in the literature, but in this case, it was preferred to use a windowed version of this algorithm.

The last two methods implemented are both tree-based, with one intended for batch processing and the other for streaming data. **Isolation Forest** builds an ensemble of Isolation Trees, which are tree structures constructed to isolate every single instance effectively. Because of susceptibility to isolation, anomalies are isolated closer to the root of the trees. IForest converges quickly with a minimal number of trees, and it requires a small sub-sampling size. As it happens for LOF, this algorithm is used in a windowed manner.[38]

There exists some incremental tree-based anomaly detectors and the one utilized in this study is **Half Space Trees** (HST). This algorithm updates its model one sample at a time requiring a constant amount of memory. This feature is particularly advantageous due to the large volume of data present in a streaming scenario. [39]

3.7. Data imputation Methods

These techniques are used for both data imputation and outlier correction and are applied once the window is complete. The first method, employed mostly on outlier correction,

is **Dropping**. Specifically, this involves discarding data points that display anomalies or contain missing values. This helps to achieve a better machine learning model used in the analysis, but it lost some informative power.

Another approach utilised is **Last Observation Carried Forward** (LOCF), where the values are propagated from the first previous valid row. This is a form of data interpolation which helps to maintain continuity.

A commonly used technique is to replace missing data points or outliers using the **Mean** of the values. In our particular case, a windowing approach was applied, so for each window and every dimension, averages were computed and put in place of missing values or outliers. This practice, however, is not the most suitable in multivariate time series as it introduces discontinuity in data.

The last method employed is **Linear Interpolation**. It is a method of curve fitting using linear polynomials. The mathematical formula to interpolate a generic point (x, y) between two points (x_1, y_1) and (x_2, y_2) is $y = y_1 + (x - x_1)\frac{(y_2 - y_1)}{(x_2 - x_1)}$. Unlike averaging, linear interpolation proves to be a particularly fitting method for time series analysis. This is attributable to its ability to maintain data continuity.

4 | Evaluation and Results

In this chapter, the methods used to evaluate the experiments outlined in the preceding chapter are presented. After that, the findings obtained from those experiments will be explained and illustrated.

4.1. Evaluation methods

The goal is to test different methods for data preparation in different data quality conditions.

Each experiment was repeated eight times to decrease the variance given by the intrinsic randomness of the used methods. Section 4.2 shows all the results obtained using graphs and tables. The graphs show the performance trend depending on the quality of the dataset. The performance trend is represented using its value of R2-score in the case of regression and F1-score in the case of classification, while the quality, on the x-axis, is represented with the percentage level of quality. The graphs feature two dashed lines labelled "Clean" and "No Action" in the legend. These two lines show the performance trend achieved on the clean dataset and the performance achieved without any data preparation actions.

In the results, dropping outliers is considered a form of outlier correction. This implies that data points detected as outliers will not appear in the regression task. Consider that, in these cases, the predictions will be biased towards correct values and will not be complete. The tables are used to perform the comparison between data streams and tabular data. Additional tables act to present the whole set of results are included in Appendix A

4.2. Results

This section presents the outcomes derived from the diverse set of experiments delineated in 2.3.2. These experiments strategically involve the injection of various types of errors,

incorporating a range of outlier detection methods and employing distinct data imputation techniques. The comprehensive nature of these experiments aims to thoroughly assess and analyze the influence of each method on the analysis performance, shedding light on their efficacy in handling different aspects of data preparation, such as outlier detection and data imputation, within the dynamic context of the conducted experiments. The elucidation of these results contributes valuable insights into the strengths and limitations of each approach, informing future decisions and refinements in the data preparation process for dynamic and evolving data streams.

4.2.1. Injecting Missing values

As outlined in the previous chapter, the first experiment aims to investigate the impact of the proposed data imputation techniques on machine learning tasks. The graph shows the ML performance metric according to the completeness dimension in the dataset. The best results are achieved by LOCF method and by linear interpolation. This outcome was anticipated since the analysed data is basically a multivariate time series. With this type of data, it is preferable to maintain a stable trend in the values. When the mean is computed in the sliding window, and then used to impute missing values, it can create some discontinuity or unconformity in the time series trend. This leads to a worse result in the ML task that has been performed. The results obtained in both AirQuality and NEWeather datasets are similar and coherent and are shown in figure 4.1 and 4.2. The main contrast seen is that in the AirQuality data, regression works better at keeping missing values rather than replacing them with the mean.

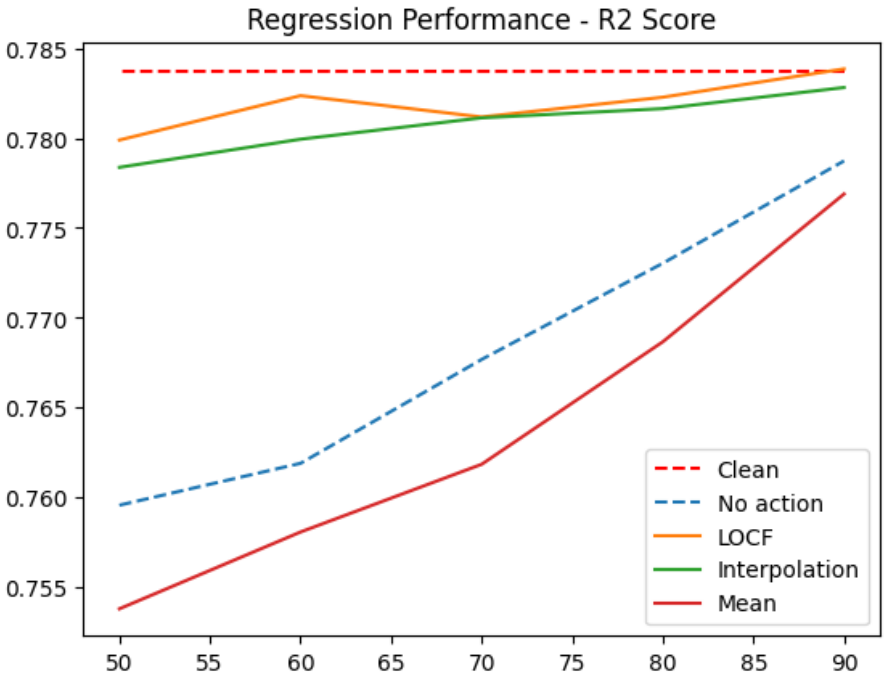


Figure 4.1: Experiment no.1 on AirQuality dataset

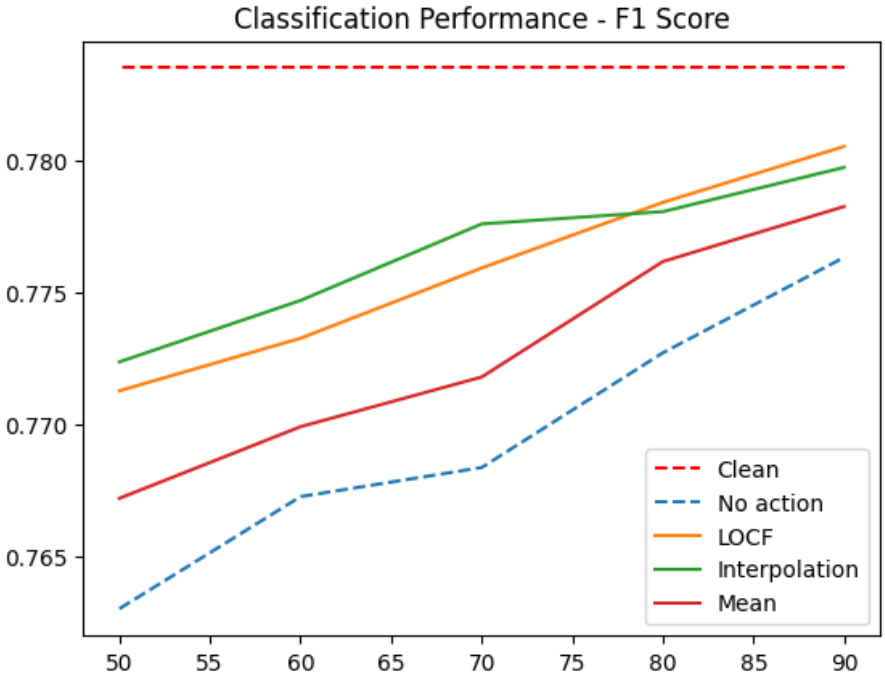


Figure 4.2: Experiment no.1 on NEWeather dataset

4.2.2. Injecting Outliers

These experiments aim to evaluate various outlier detection techniques and compare the results using two separate evaluations. The initial assessment aims to find the best outlier detection method while the second investigation focuses on regression performance after detecting and nullifying outliers. The goodness of the methods is computed using the F1-score comparing indices of detected outliers with the indices of injected outliers.

In this particular instance, the most effective method to detect outliers is Local Outlier Factor. It is worth noting that the distance-based approach outperforms both Isolation Forest and HST, which are tree-based, for both batch and streaming data.

The Z-score method is unsuitable due to its susceptibility to variance, as it may fail to identify any outlier within the NEWeather dataset and produce unsatisfactory results within the AirQuality dataset. In this case, it can be noted that it is easier for him to recognise more outliers when the accuracy of the dataset is higher. Conversely, the low accuracy of data will lead to a higher standard deviation in the values and subsequently result in the recognition of fewer outliers.

Isolation Forest behaves differently compared to the other approaches. It detects almost the same quantity of outliers with varying levels of accuracy. In AirQuality dataset, this leads to an unusual response. The quantity of detected anomalies almost matches the quantity of anomalies inserted at an accuracy rate of 80%. This explains the peak shown in the graph. This occurs in the NEWeather dataset as well, however, the number of identified outliers corresponds with the number of outliers at a 90% level of accuracy as demonstrated by the trend in the graph.

The graphs clearly show that Half Space Trees exhibits an inverse trend in comparison with other methods. This algorithm is able to correctly recognise more outliers when the quality of the dataset is poor. This noteworthy result was initially reported in other works, including the original paper in which the algorithm is presented [39]. In our specific instance, a possible cause of this behaviour is the first stage of the model training. During the initial 1000 iterations, the streaming algorithm recognises numerous data points as anomalies thereby affecting the obtained results. A lower dataset accuracy corresponds to an increase in outliers detected, which may lessen the relevance of the initial errors in the outcome. At this point, the unrecognised anomalies in the first part of the dataset will have a greater influence on the results when the dataset is cleaner and fewer outliers are present.

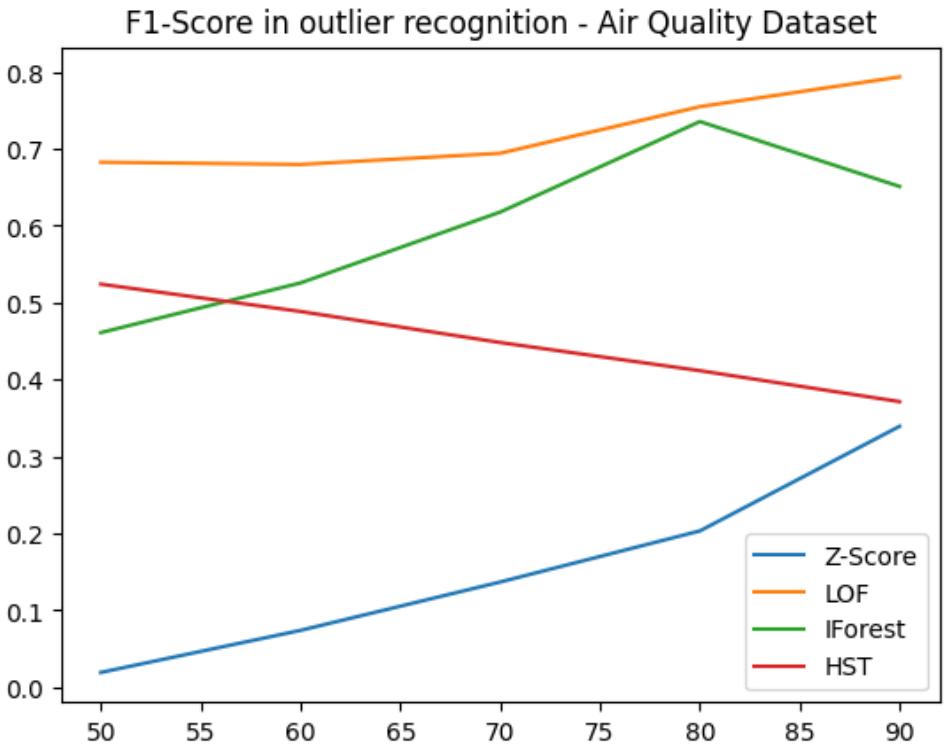


Figure 4.3: Experiment no.2 on AirQuality dataset

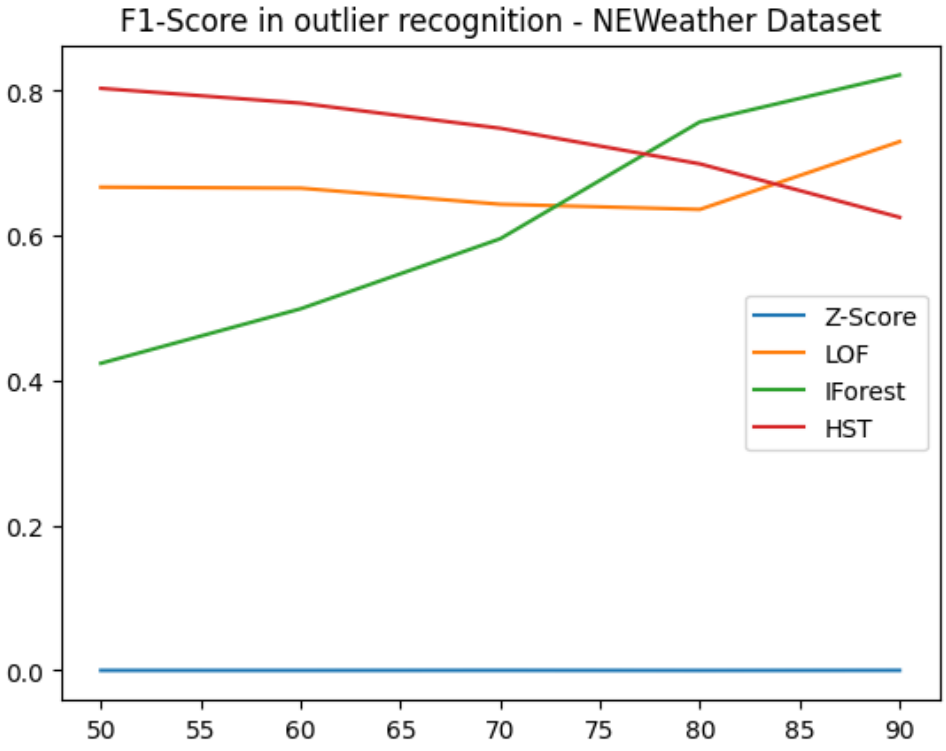


Figure 4.4: Experiment no.2 on AirQuality dataset

These results are partially reflected in the ML tasks. Using both Random Forest Regressor and Random Forest Classifier, it is noticeable that it is better to maintain all the outliers in the dataset rather than nullify them. In Figure 4.5, the blue dashed line is above each other method, for each evaluated data quality percentage. This is caused by the standardization of all the outliers. In fact, putting all the detected outliers equal to a standard value does not help to build a proper regression model. This is observable since the Z-score, which in the previous experiment detected fewer outliers than other methods, has the best performance along the analysed techniques.

The same results occur when using the NEWeather dataset. In this case, the performance achieved after Z-score follows completely the "No Action" dashed line. This happens because it does not detect any outliers. The main difference between the two datasets occurs when using Isolation forest. When utilizing the AirQuality dataset the regression performance remains consistent, only deteriorating when a tenth of outliers are introduced. This occurs as the outlier detection method recognises a greater number of anomalies than exist. In the NEWeather dataset, the detection of outliers in nearly 10% of the rows contributes to improved performance at this level of accuracy.

To summarise, all the results in the ML tasks, besides the already illustrated IForest case, improve as accuracy improves. This is an expected outcome since with fewer outliers injected into the dataset we could expect a more efficient ML model.

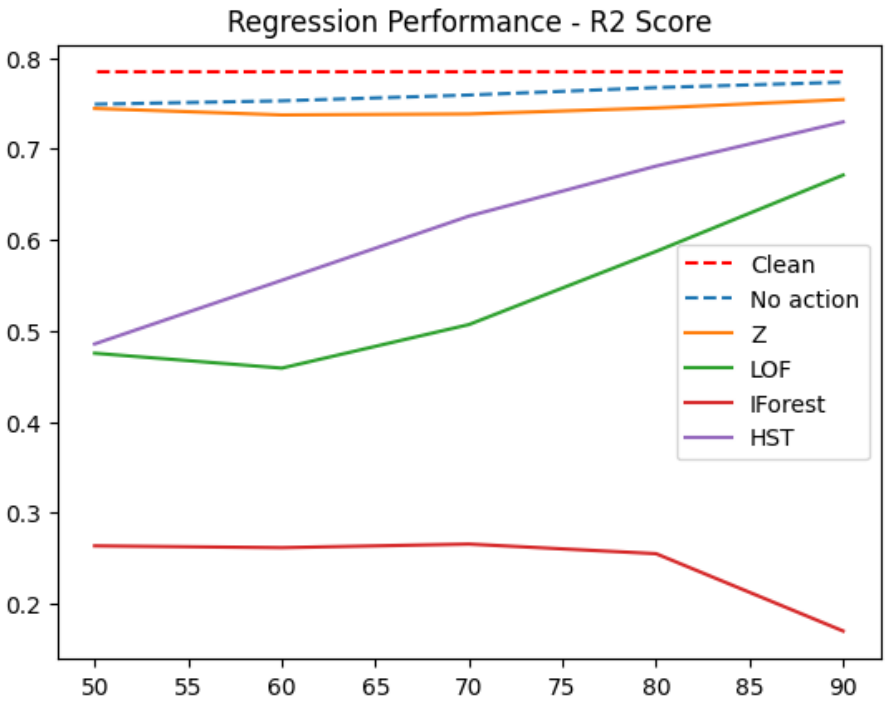


Figure 4.5: Experiment no.3 on AirQuality dataset

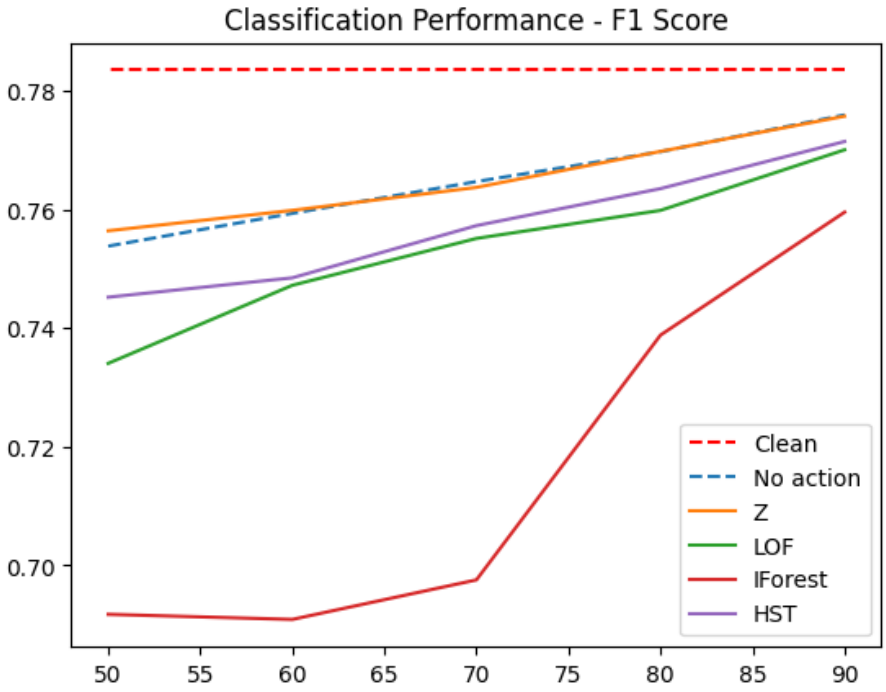


Figure 4.6: Experiment no.3 on NEWeather dataset

The third trial, in which outliers are detected and subsequently corrected, discloses an unexpected outcome. The experimentation concludes that better results in regression or classification are achieved by avoiding the correction of anomalies. Actually, the only way to improve the results appears to be eliminating detected outliers. In both datasets, every outlier correction method worsened the ML performance. A following test, performed with KNN regressor and KNN classifier, revealed that this issue is due to the machine learning models applied. In fact, both random forest regressor and classifier are robust to outliers.

When using the AirQuality dataset dropping anomalies seems to improve the outcomes only when they are detected with Local Outlier Factor or with Z-Score. Cancelling outliers on the NEWeather dataset, instead, improves the results even when Half Space Trees or Isolation Forest are used. In the last case, dropping the outliers spotted by Isolation Forest provides a performance better than what is achieved with the clean dataset. This is not particularly meaningful since it must considered that for many data points there will not be any prediction, and consequently, the algorithm will have less informative power.

The behaviour of the individual combinations leads back to the trends of the previous experiments, with a generic positive trend in most of the applied methods, both in NEWeather and in AirQuality datasets. The most out-of-the-line result is obtained

for the AirQuality dataset when using IForest as the outlier detection method. In this particular circumstance, using the mean imputation to correct the outliers leads to a negative R2-score during the regression validation. The other approaches used to correct outliers bring a negative trend in the graphs. This means that the R2-score gets worse when the dataset has a better quality. This trend can be caused by an overfitting due to the presence of outliers in the data stream. Isolation Forest also provides peculiar results when the analysed dataset is NEWeather. Here, there is further evidence of the results of the second task. All the correction methods provide a steady trend in the graph, and assuming that IForest found a similar quantity of outliers, independently from the percentage of injected outliers, it makes sense.

In this experiment, as happens in the previous, Z-score hardly recognises any outliers in the NEWeather dataset, so all the methods yield similar outcomes. The fluctuation is attributable to the randomness factor in the classification method. The elimination of outliers detected by Half Space Trees in the same data stream results in a particular but comprehensible conclusion. Considering that HST works better the more outliers there are, and considering that the next step is removing almost 50% of the dataset, it is not so weird to have such a higher value of F1-Score

As previously stated, this experiment was replicated using K-Nearest Neighbour. The outcomes reveal corresponding trends to those obtained with Random Forest. Nonetheless, in this instance, outlier correction proves positively influential on performance. Figure 4.9 displays the outcomes achieved through KNN regression on the AirQuality dataset.

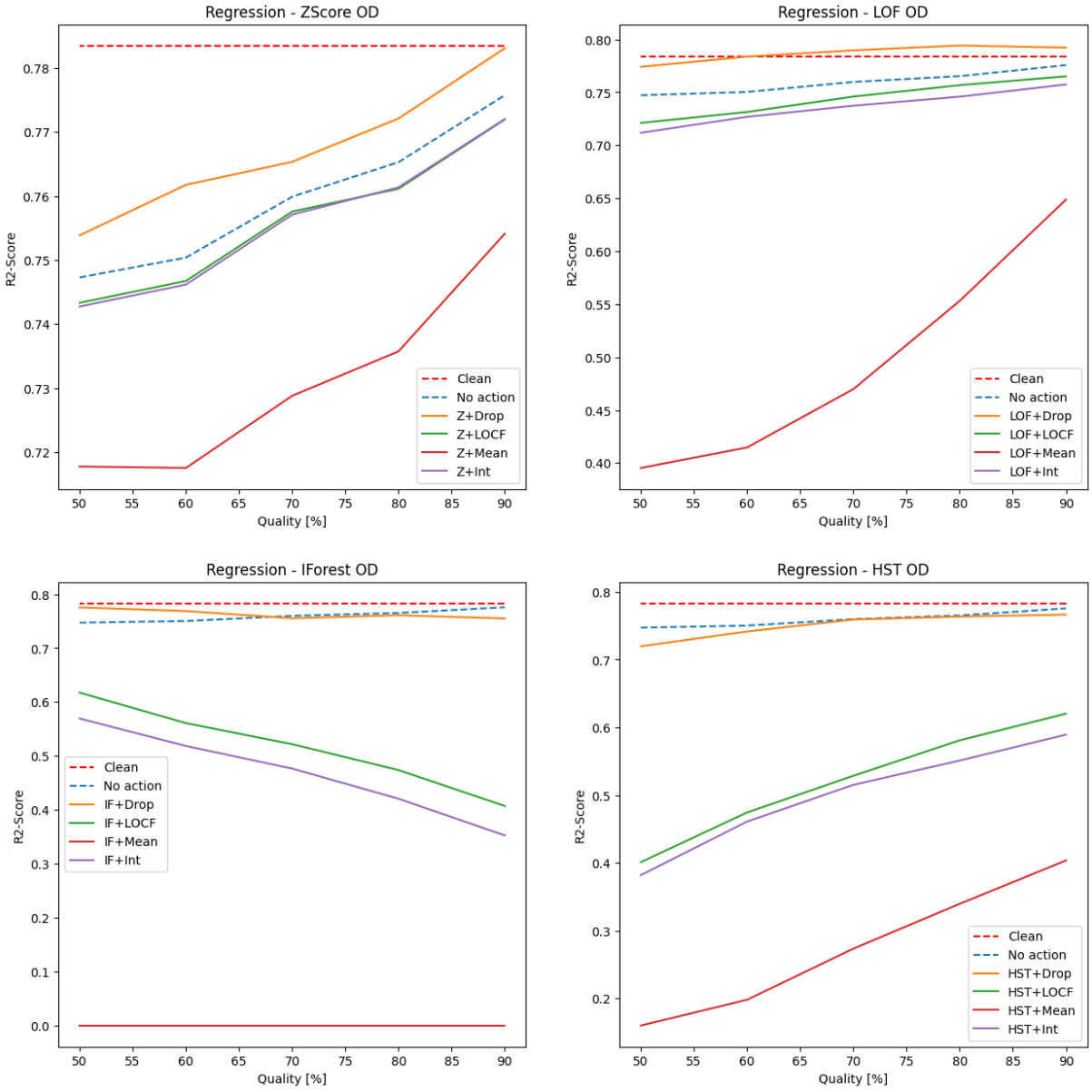


Figure 4.7: Experiment no.4 on AirQuality dataset

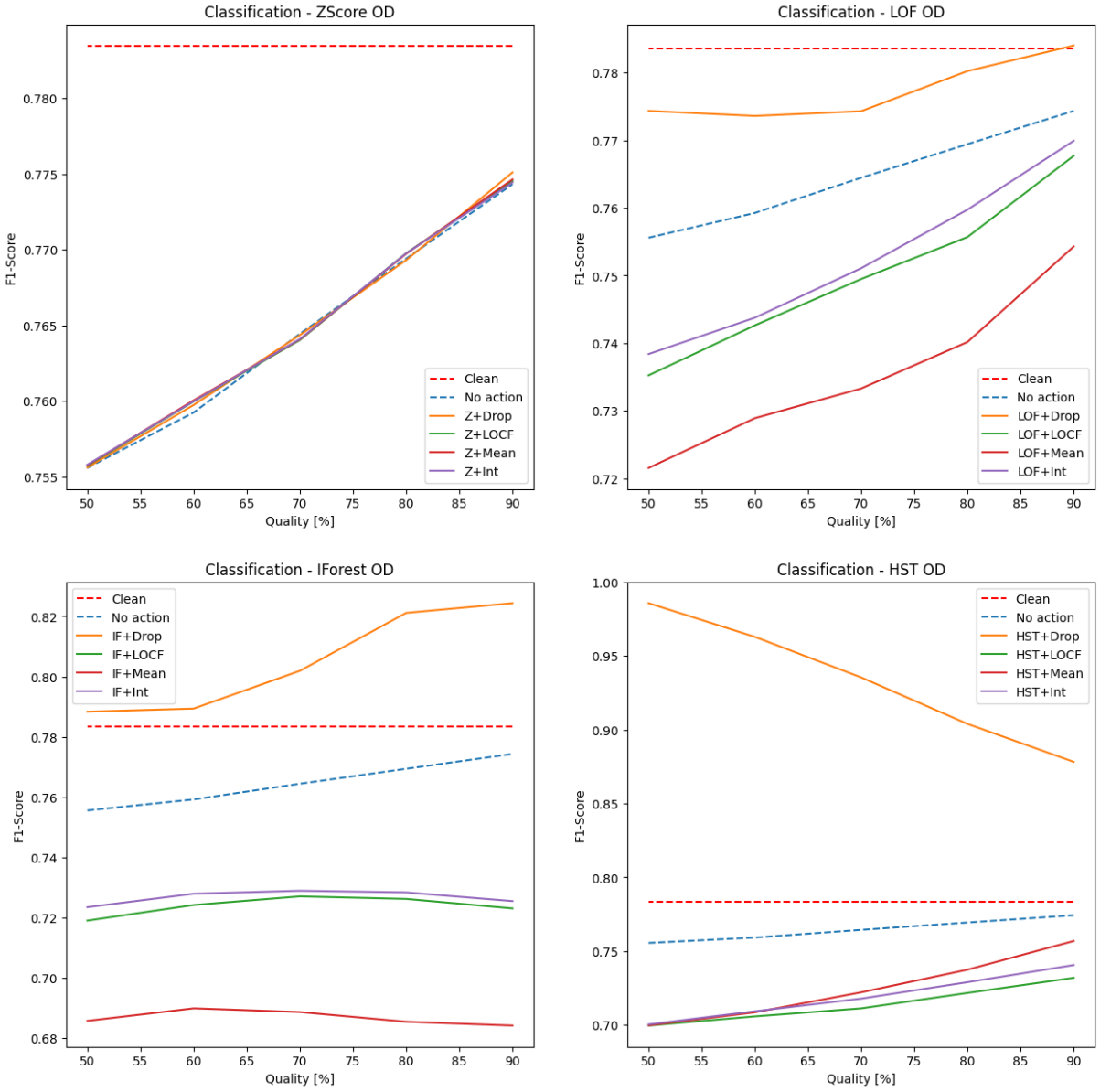


Figure 4.8: Experiment no.4 on NEWeather dataset

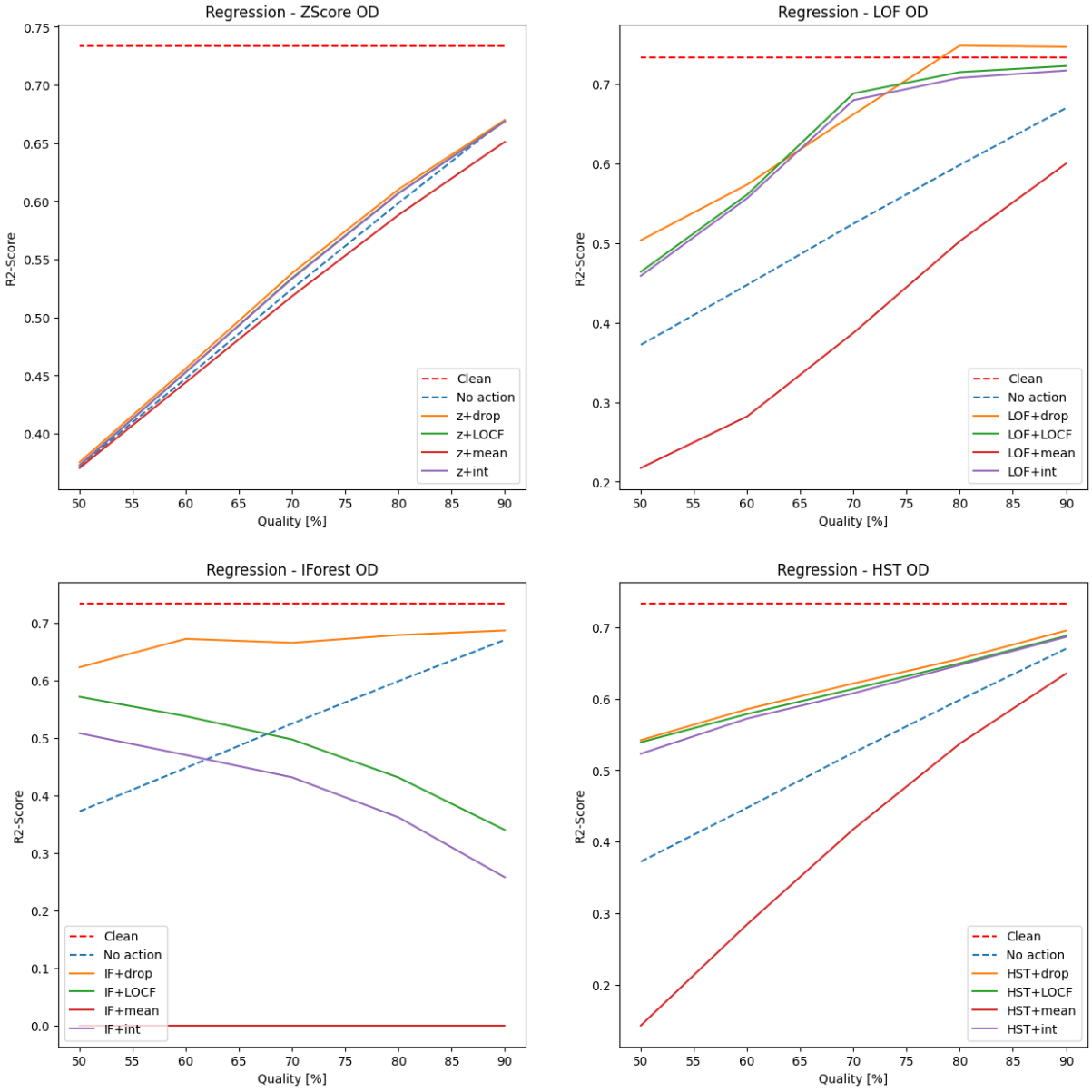


Figure 4.9: Experiment no.4 on AirQuality dataset - KNN Regressor

4.2.3. Injecting Outliers and Missing Values

The final experiments regarding outlier detection and correction and data imputation involve the injection of both outliers and missing values in the used dataset. Following this injection, both tasks are carried out to prepare and clean the dataset before analysis. The main objective of these investigations is to conduct an experimental test to determine whether there is an alteration when the order of the operations is reversed. Differences can occur because when outlier detection is performed first, the OD methods consider the rows with missing values and can mark them as outliers. In the other case, missing

values imputation will include outliers, which may lead to additional errors. Only 12 combinations of outlier detection, outlier correction and data imputation methods are investigated for lack of time, and the selection is based on the results achieved in the previous experiments. Dropping missing values would have improved completeness, but it would have led to a loss of informative power, so it was not considered, as outliers are already dropped in many cases.

In the first assessment, outlier correction is performed first followed by data imputation. In the AirQuality dataset, when using Z-Score, all the tried methods return satisfying outcomes, with each of these methods improving the base performance. Dropping the detected outliers helps to achieve a better performance with respect to the clean dataset. This happens for every outlier detection method selected, but it must be considered that dropping outliers implies a loss of information. The trends observed with LOF and HST in this experiment are not attributed to outlier correction but to the imputation of missing values. It is evident that the trends observed in this specific experiment resemble the sum of the trends achieved in the first and in the third experiment.

When using the NEWeather dataset, while Z-Score does not detect any outliers as shown before, the other methods achieve expected results, in which dropping outliers is preferable to correct them using LOCF, and LOCF has very similar results to interpolation when they are used to impute the missing values.

To summarise, the optimal results are achieved with the utilisation of Local Outlier Factor for identifying outliers, followed by their removal and linear interpolation of missing values. When outliers are not removed to retain, the most effective approach includes the Z-score method for outlier detection and substituting them by propagating the preceding value. The choice between using LOCF or linear interpolation to fill in missing values holds no significance, as their effectiveness is essentially the same. These results are illustrated in Figure 4.10

In the last trial, missing values imputation was performed first, followed by outlier detection and correction. This sequence leads to a hypothetical error, as data imputation is completed before correcting any dataset outlier. This should create a bias towards outliers that eventually propagate into the clean data points containing missing values. Empirical results suggest that the order of operations has a very low impact on the overall outcome. The obtained results are almost identical to those from the previous step, with a slight advantage towards them. This implies that the optimal combinations remain unchanged. The full results of this experiment are shown in Figure 4.11.

Figure 4.12 compares regression performance when the order is changing, using the most significant method combination.

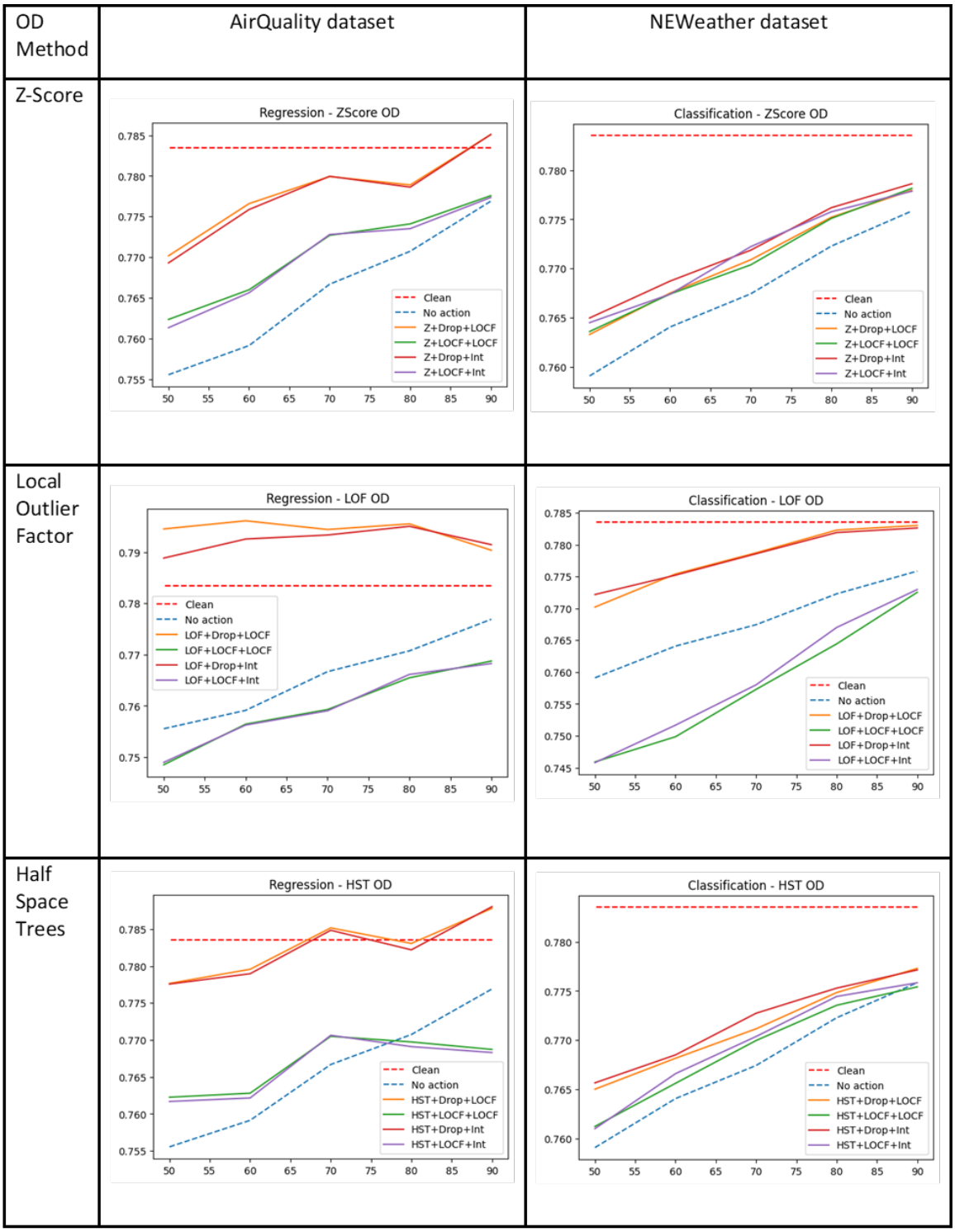


Figure 4.10: Experiment No.5

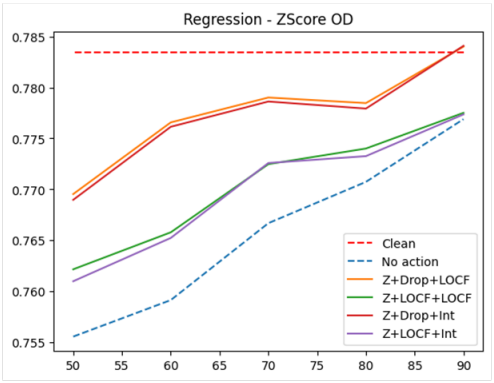
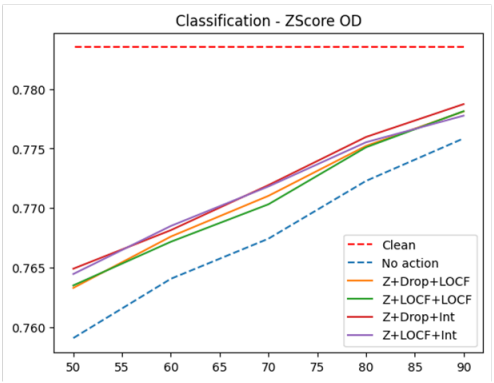
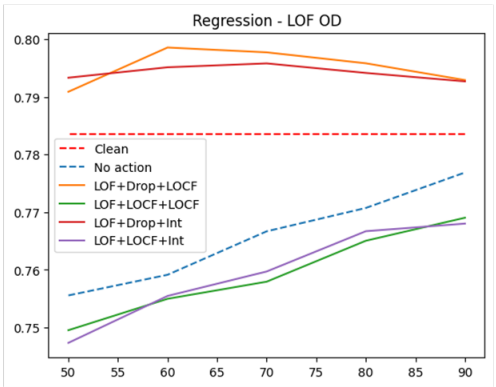
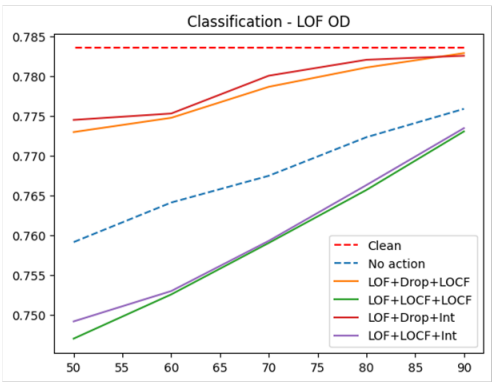
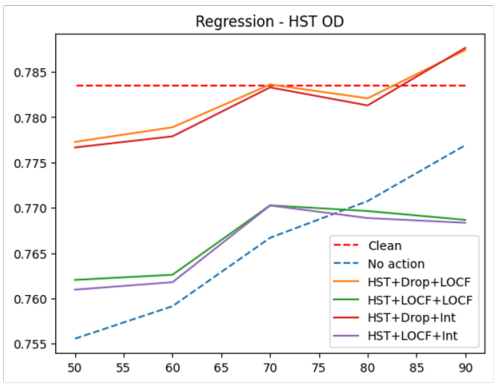
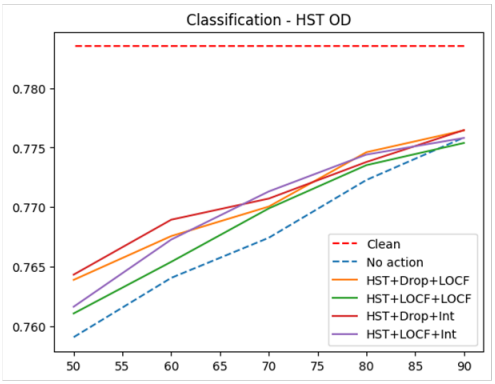
OD Method	AirQuality dataset	NEWeather dataset
Z-Score		
Local Outlier Factor		
Half Space Trees		

Figure 4.11: Experiment No.6

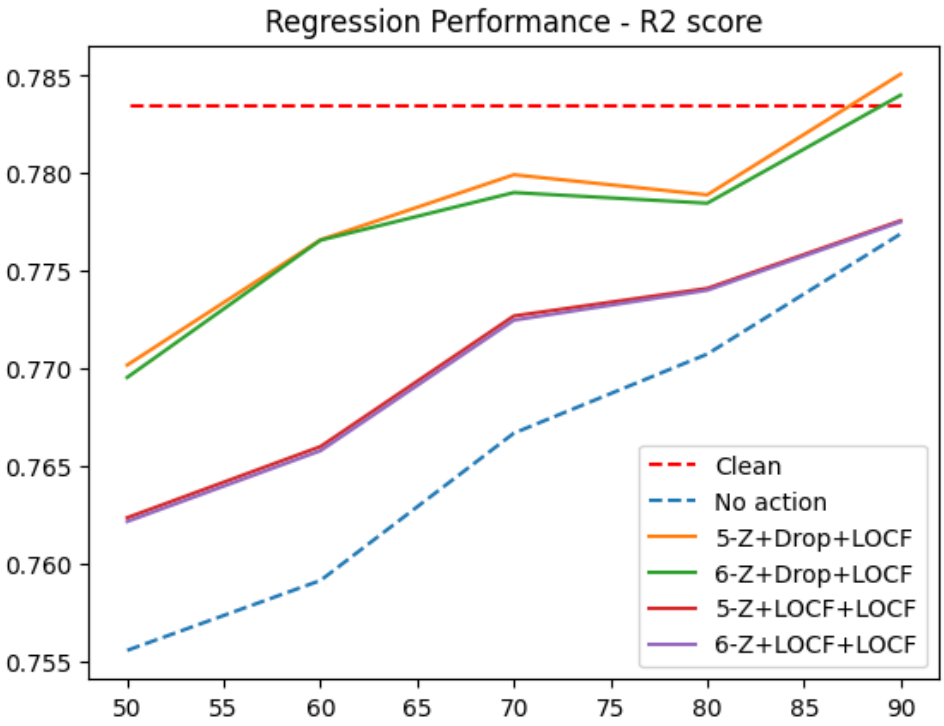


Figure 4.12: Comparison between Experiment no.5 and no.6.

4.3. Differences between streaming and tabular data

This study aims to compare the efficiency of data preparation in data stream and in tabular data. Section 2.3 presents the conducted experiments. These experiments are analogous to those reported previously.

The accuracy attained by the machine learning when using the clean dataset $\overline{p}$, is compared with the accuracy p_{ij} achieved within the injected dataset d_i, post data preparation action dpa_j. The percentage change is calculated as

$$d\%_{ij} = \frac{\overline{p} - p_{ij}}{\overline{p}} \times 100\%.$$

The comparison is performed by calculating the difference between each d_{ij} attained from the data stream and each d_{ij} attained from the Electrical dataset

The graphics featured below indicate the comparison after every distinct experiment. The findings indicate that the majority of the attempted techniques are more effective in data streams than with a tabular dataset. This result is more evident in the regression task than in the classification task. The only exception is the mean imputation which has

a better impact on the Electrical dataset. As previously explained, this is an expected outcome since the mean introduces discontinuities in data streams.

The second experiment demonstrates that Local Outlier Factor and Isolation Forest perform better in data streams when using a windowed approach, in contrast with Z-score, which performs much better when computed in batch rather than incrementally. These findings could be influenced by the intrinsic properties of the various datasets and are shown in the figures below.

The fourth experiment expresses the most meaningful results. Figure 4.16 shows a comparison of the regression performances. The values were calculated as described previously. The green cells indicate an advantage for the data stream, while the red cells favour tabular data. In addition to the poor performance of the mean, the combination of Isolation Forest and linear interpolation did not perform well. These results are analogous to what happens in the classification task, as shown in Figure 4.17. The only difference is due to Z-score, which in the NEWeather dataset is not capable of recognising any outlier.

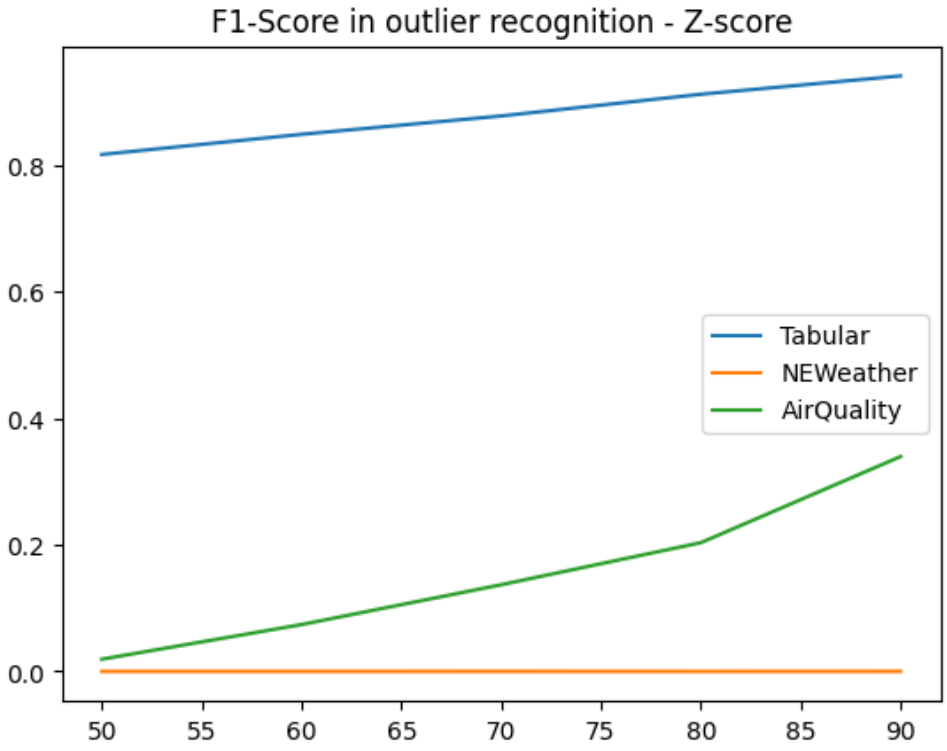


Figure 4.13: Comparison in Z-score outlier recognition

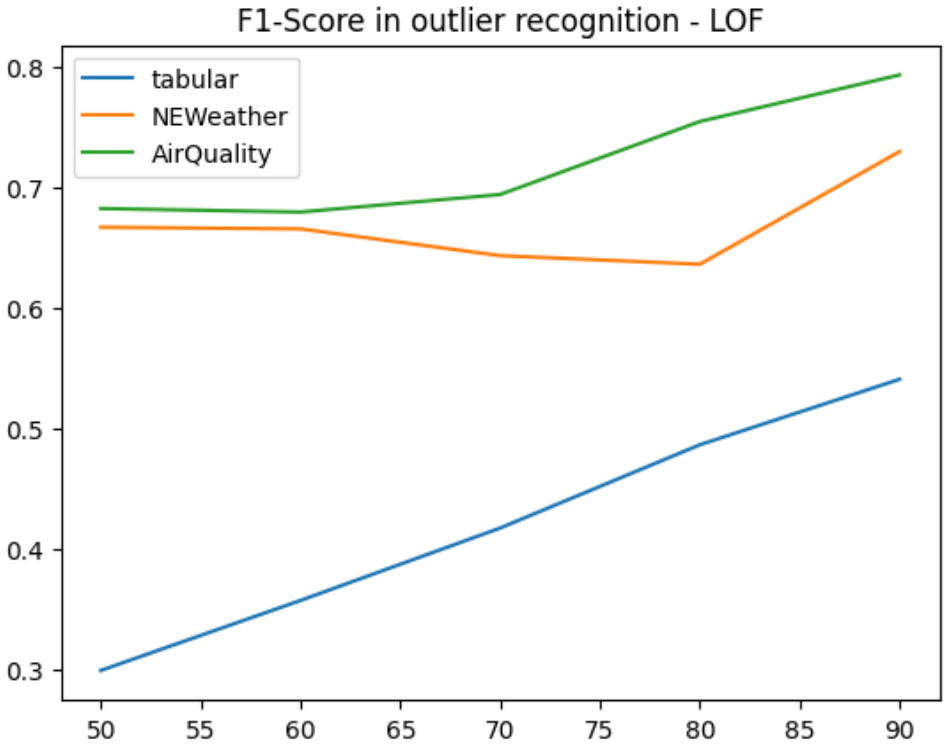


Figure 4.14: Comparison in LOF outlier recognition

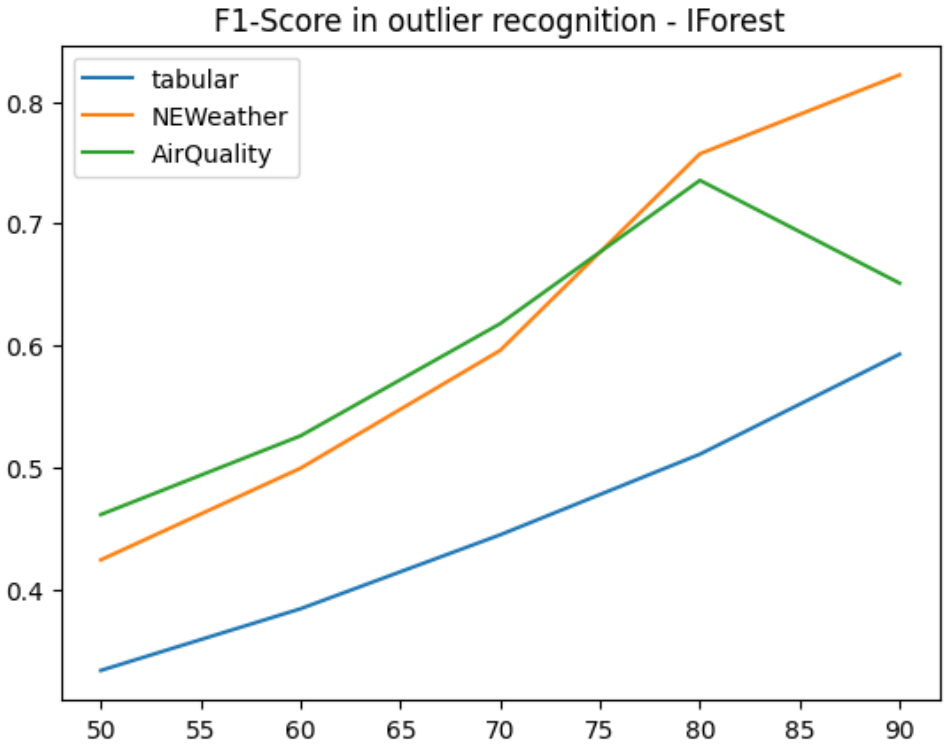


Figure 4.15: Comparison in Isolation Forest outlier recognition

Outlier	Imputation	50%	60%	70%	80%	90%
none	none	13.12	10.14	9.16	6.61	4.11
lof	drop	13.99	12.81	10.03	7.2	3.95
z	drop	5.51	4.18	1.92	0.45	0.75
z	interpolation	50.83	43.28	34.87	25.47	14.34
lof	interpolation	18.16	18.03	15.54	11.46	6.3
iforest	drop	14.44	11.48	6.16	3.65	-0.63
z	mean	32.38	26.11	19.7	12.32	6.23
lof	mean	-27.2	-27.02	-23.38	-17.32	-10.31
iforest	interpolation	2.93	-6.4	-15.85	-28.13	-44.65
iforest	mean	-75.91	-78.22	-82.28	-87.12	-92.6

Figure 4.16: Comparison in Regression Performance

Outlier	Imputation	50%	60%	70%	80%	90%
none	none	2.58	1.67	1.64	1.18	0.08
iforest	drop	6.22	4.99	5.63	6.4	5.98
lof	drop	4.26	2.78	1.56	1.38	0.71
z	drop	-0.17	-1.03	-1.26	-1.31	-1.2
z	mean	9.26	7.82	5.61	3.72	1.81
z	interpolation	11.34	9.5	7.04	4.93	2.21
lof	interpolation	2.12	1.63	1.4	1.16	0.42
lof	mean	-0.13	-0.39	-1.22	-1.75	-1.53
iforest	interpolation	1.21	0.26	-0.84	-2.57	-4.98
iforest	mean	-3.9	-4.78	-6.32	-8.41	-10.47

Figure 4.17: Comparison in Classification Performance

5 | Conclusions and future developments

The increasing enthusiasm for data analysis and data streams drove the development of this thesis. This interest in data analysis has prompted companies to make numerous data-driven decisions. Emphasizing data quality is crucial to ensure the most accurate analysis possible and prevent the propagation of errors present in the data to the decision-making process.

The importance of data quality is evident not only in traditional datasets but also in the realm of data streams. Given their dynamic and ever-evolving nature, data streams demand methods that skillfully balance the need for velocity with optimal system performance.

This thesis works on top of an already existing framework, designed to propose a set of data preparation actions from a knowledge base before engaging in data analysis tasks. The goal is to augment and refine this existing framework, extending its support to the challenges posed by data streams.

The executed modifications are present throughout the pipeline. The initial adaption focuses on data profiling. A tool has been developed to facilitate continuous stream profiling. This enables real-time insight into the evolving characteristics of the data stream.

The outcome of the stream profiling is successively used to assess, in real-time, data quality according to some predefined dimensions.

The ultimate enhancement to the architecture is associated with the knowledge base, where data preparation actions adapted to data streams have been incorporated. This addition follows an experimental phase in which various techniques were tested, and their impact on data analysis was thoroughly evaluated.

The results reveal a deviation from the expected outcomes. The anticipated outcome was that imputing outliers using mean is misleading for the model because it causes

discontinuity in the data. The most surprising finding is that retaining outliers in the dataset proves more beneficial than correcting them through substitution, contrary to the initially anticipated outcome. The only method that seems to improve the performance is dropping outliers, which implies a loss of informative power. A subsequent test revealed that these results can be attributed to the utilization of an outlier-resistant model, such as Random Forest. In contrast, when employing a simpler method like KNN, the outcomes align more closely with our expectations.

In summary, the most suitable outlier detection method appears to be the Local Outlier Factor used with a window approach, while the optimal techniques for imputation of missing values are Last Observation Carried Forward and Linear Interpolation.

The most pressing task ahead is to test the experiments using the KNN model to validate the accuracy of our previous assumptions. Looking ahead, the future development of this work can cover various aspects of the architecture. Firstly, broadening data sources to include unstructured data streams or univariate time series would significantly enhance the system's versatility. Additionally, introducing novel data profiling techniques, such as incremental functional dependency discovery, as an alternative to the current windowed approach, can further refine our understanding of the data.

Furthermore, investigating novel techniques for detecting outliers through the use of deep learning, and employing approaches tailored expressly for univariate time series employing ARIMA model-based strategies presents possibilities for improvement. Moreover, there is potential for advancing data imputation methods by incorporating deep learning approaches, contributing to a more comprehensive and robust data analysis and preparation framework.

Bibliography

[1] Alessandro Margara and Tilmann Rabl. Definition of data streams. In *Encyclopedia of Big Data Technologies*, pages 1–4. Springer International Publishing, 2018. doi: 10.1007/978-3-319-63962-8_188-1. URL https://doi.org/10.1007/978-3-319-63962-8_188-1.

[2] Taiwo Kolajo, Olawande Daramola, and Ayodele Adebiyi. Big data stream analysis: a systematic literature review. *Journal of Big Data*, 6(1), June 2019. doi: 10.1186/s40537-019-0210-7. URL https://doi.org/10.1186/s40537-019-0210-7.

[3] Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. Data stream management: A brave new world. In *Data-Centric Systems and Applications*, pages 1–9. Springer Berlin Heidelberg, 2016. doi: 10.1007/978-3-540-28608-0_1. URL https://doi.org/10.1007/978-3-540-28608-0_1.

[4] Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, June 2004. doi: 10.1145/1055558.1055598. URL https://doi.org/10.1145/1055558.1055598.

[5] Peter J. Haas. Data-stream sampling: Basic techniques and results. In *Data-Centric Systems and Applications*, pages 13–44. Springer Berlin Heidelberg, 2016. doi: 10.1007/978-3-540-28608-0_2. URL https://doi.org/10.1007/978-3-540-28608-0_2.

[6] Lukasz Golab and M. Tamer Özsu. Issues in data stream management. *ACM SIGMOD Record*, 32(2):5–14, June 2003. doi: 10.1145/776985.776986. URL https://doi.org/10.1145/776985.776986.

[7] Yair Wand and Richard Y. Wang. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11):86–95, November 1996. doi: 10.1145/240455.240479. URL https://doi.org/10.1145/240455.240479.

[8] Giri Kumar Tayi and Donald P. Ballou. Examining data quality. *Communications*

of the ACM, 41(2):54–57, February 1998. doi: 10.1145/269012.269021. URL https://doi.org/10.1145/269012.269021.

[9] Richard Y Wang and Diane M Strong. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33, 1996.

[10] Carlo Batini and Monica Scannapieco. *Data and Information Quality*. Springer International Publishing, 2016. doi: 10.1007/978-3-319-24106-7. URL https://doi.org/10.1007/978-3-319-24106-7.

[11] A. Klein and W. Lehner. Representing data quality in sensor data streaming environments. *Journal of Data and Information Quality*, 1(2):1–28, September 2009. doi: 10.1145/1577840.1577845. URL https://doi.org/10.1145/1577840.1577845.

[12] Fernando Gualo, Moisés Rodríguez, Javier Verdugo, Ismael Caballero, and Mario Piattini. Data quality certification using iso/iec 25012: Industrial experiences, 2021.

[13] Sandra Geisler, Sven Weber, and Christoph Quix. Ontology-based data quality framework for data stream applications. In *ICIQ*, 2011.

[14] Lisa Ehrlinger, Elisa Rusz, and Wolfram Wöß. A survey of data quality measurement and monitoring tools, 2019.

[15] Ricardo Perez-Castillo, Ana G. Carretero, Ismael Caballero, Moises Rodriguez, Mario Piattini, Alejandro Mate, Sunho Kim, and Dongwoo Lee. Daqua-mass: An iso 8000-61 based data quality management methodology for sensor data. *Sensors*, 18(9), 2018. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/18/9/3105.

[16] Lina Zhang, Dongwon Jeong, and Sukhoon Lee. Data quality management in the internet of things. *Sensors*, 21(17):5834, August 2021. doi: 10.3390/s21175834. URL https://doi.org/10.3390/s21175834.

[17] Miaomiao Yu, Chunjie Wu, and Fugee Tsung. Monitoring the data quality of data streams using a two-step control scheme. *IISE Transactions*, 51(9):985–998, February 2019. doi: 10.1080/24725854.2018.1530487. URL https://doi.org/10.1080/24725854.2018.1530487.

[18] Felix Naumann. Data profiling revisited. *ACM SIGMOD Record*, 42(4):40–49, February 2014. doi: 10.1145/2590989.2590995. URL https://doi.org/10.1145/2590989.2590995.

[19] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. Profiling relational

data: a survey. *The VLDB Journal*, 24(4):557–581, June 2015. doi: 10.1007/s00778-015-0389-y. URL https://doi.org/10.1007/s00778-015-0389-y.

[20] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29, 1996.

[21] Phillip B Gibbons. Distinct-values estimation over data streams. In *Data Stream Management: Processing High-Speed Data Streams*, pages 121–147. Springer, 2016.

[22] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005. ISSN 0196-6774. doi: https://doi.org/10.1016/j.jalgor.2003.12.001. URL https://www.sciencedirect.com/science/article/pii/S0196677403001913.

[23] Moses Charikar. Top- k k frequent item maintenance over streams. In *Data-Centric Systems and Applications*, pages 103–119. Springer Berlin Heidelberg, 2016. doi: 10.1007/978-3-540-28608-0_5. URL https://doi.org/10.1007/978-3-540-28608-0_5.

[24] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record*, 30(2):58–66, May 2001. doi: 10.1145/376284.375670. URL https://doi.org/10.1145/376284.375670.

[25] Zhiwei Chen and Aoqian Zhang. A survey of approximate quantile computation on large-scale data. *IEEE Access*, 8:34585–34597, 2020. doi: 10.1109/access.2020.2974919. URL https://doi.org/10.1109/access.2020.2974919.

[26] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, page 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1558601538.

[27] Philipp Schirmer, Thorsten Papenbrock, Sebastian Kruse, Felix Naumann, Dennis Hempfing, Torben Mayer, and Daniel Neuschäfer-Rube. Dynfd: Functional dependency discovery in dynamic datasets. In *International Conference on Extending Database Technology*, 2019. URL https://api.semanticscholar.org/CorpusID:81987855.

[28] Mazhar Hameed and Felix Naumann. Data preparation. *ACM SIGMOD Record*, 49(3):18–29, December 2020. doi: 10.1145/3444831.3444835. URL https://doi.org/10.1145/3444831.3444835.

[29] Sergio Ramírez-Gallego, Bartosz Krawczyk, Salvador García, Michał Woźniak, and Francisco Herrera. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39–57, May 2017. doi: 10.1016/j.neucom.2017.01.078. URL https://doi.org/10.1016/j.neucom.2017.01.078.

[30] Vincenzo Gulisano, Dimitris Palyvos-Giannas, Bastian Havers, and Marina Papatriantafilou. The role of event-time order in data streaming analysis. In *Proceedings of the 14th ACM International Conference on Distributed and Event-based Systems*. ACM, July 2020. doi: 10.1145/3401025.3404088. URL https://doi.org/10.1145/3401025.3404088.

[31] Feature Deep Dive: Watermarking in Apache Spark Structured Streaming — databricks.com. https://www.databricks.com/blog/2022/08/22/feature-deep-dive-watermarking-apache-spark-structured-streaming.html. [Accessed 24-10-2023].

[32] Rajalakshmi Krishnamurthi, Adarsh Kumar, Dhanalekshmi Gopinathan, Anand Nayyar, and Basit Qureshi. An overview of IoT sensor data processing, fusion, and analysis techniques. *Sensors*, 20(21):6076, October 2020. doi: 10.3390/s20216076. URL https://doi.org/10.3390/s20216076.

[33] Manu Siddhartha. Beijing multi-site air-quality data set, Nov 2019. URL https://www.kaggle.com/datasets/sid321axn/beijing-multisite-airquality-data-set.

[34] Gregory Ditzler and Robi Polikar. Incremental learning of concept drift from streaming imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 25: 2283–2301, 10 2013. doi: 10.1109/TKDE.2012.136.

[35] URL https://archive.ics.uci.edu/dataset/471/electrical+grid+stability+simulated+data.

[36] Zohar Karnin, Kevin Lang, and Edo Liberty. Optimal quantile approximation in streams, 2016.

[37] Tim Roughgarden and Gregory Valiant. URL https://theory.stanford.edu/~tim/s17/l/l2.pdf.

[38] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008. doi: 10.1109/ICDM.2008.17.

[39] Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. Fast anomaly detection for

streaming data. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, page 1511–1516. AAAI Press, 2011. ISBN 9781577355144.

A | Appendix A

This appendix presents the results of each experiment detailed in Section 2.3. An elucidation of the results and the evaluation methods is provided in Chapter 4. Each table corresponds to a specific experiment and dataset. The rows depict the method or combination of methods used, while the columns represent the dataset's quality percentage. Each cell contains the percentage change between the performance under those conditions and the performance achieved with the clean dataset. The performance metrics include R2-Score for regression and F1-Score for classification.

Imputation	50%	60%	70%	80%	90%
No Action	-3.08	-2.78	-2.04	-1.36	-0.63
Drop	3.47	3.38	3.16	2.56	1.21
LOCF	-0.48	-0.17	-0.32	-0.18	0.02
Interpolation	-0.68	-0.48	-0.33	-0.26	-0.11
Mean	-3.81	-3.27	-2.79	-1.92	-0.87

Experiment No.1 - AirQuality Dataset

Imputation	50%	60%	70%	80%	90%
No Action	-2.61	-2.07	-1.93	-1.38	-0.91
LOCF	-1.56	-1.31	-0.97	-0.65	-0.38
Interpolation	-1.42	-1.12	-0.75	-0.69	-0.48
Drop	-1.1	-1.15	-1.12	-0.47	-0.64
Mean	-2.08	-1.73	-1.49	-0.94	-0.67

Experiment No.1 - NEWeather Dataset

Method	50%	60%	70%	80%	90%
HST	0.5241	0.4885	0.4482	0.4116	0.3713
IForest	0.461	0.5257	0.6178	0.7354	0.651
LOF	0.6826	0.6796	0.6942	0.7548	0.7934
Z	0.0193	0.074	0.1369	0.2033	0.3393

Experiment No.2 - AirQuality Dataset

Method	50%	60%	70%	80%	90%
HST	0.8034	0.783	0.7483	0.6992	0.6254
IForest	0.4239	0.4991	0.5958	0.7572	0.8219
LOF	0.667	0.6657	0.6434	0.6363	0.7299
Z	0.0001	0.0	0.0001	0.0	0.0001

Experiment No.2 - NEWeather Dataset

Outlier	50%	60%	70%	80%	90%
No Action	-4.44	-3.96	-3.15	-2.11	-1.35
Z	-5.04	-5.94	-5.8	-4.97	-3.79
HST	-38.05	-29.13	-20.13	-13.11	-6.92
LOF	-39.35	-41.44	-35.34	-25.08	-14.37
IForest	-66.36	-66.62	-66.11	-67.46	-78.31

Experiment No.3 - AirQuality Dataset

Outlier	50%	60%	70%	80%	90%
No Action	-3.78	-3.08	-2.4	-1.75	-0.96
Z	-3.46	-3.02	-2.52	-1.75	-0.99
HST	-4.88	-4.47	-3.34	-2.55	-1.53
LOF	-6.31	-4.63	-3.62	-3.02	-1.71
IForest	-11.71	-11.82	-10.97	-5.7	-3.06

Experiment No.3 - NEWeather Dataset

Outlier	Imputation	50%	60%	70%	80%	90%
No Action	No Action	-4.62	-4.22	-3.01	-2.32	-0.99
LOF	Drop	-1.2	0.04	0.79	1.37	1.11
Z	Drop	-3.78	-2.78	-2.32	-1.46	-0.05
Z	Interpolation	-5.2	-4.76	-3.37	-2.82	-1.46
Z	LOCF	-5.13	-4.69	-3.31	-2.85	-1.47
HST	Drop	-8.14	-5.34	-3.09	-2.54	-2.16
LOF	LOCF	-7.95	-6.64	-4.78	-3.41	-2.35
LOF	Interpolation	-9.14	-7.22	-5.88	-4.79	-3.32
IForest	Drop	-1.02	-1.88	-3.58	-2.88	-3.63
Z	Mean	-8.39	-8.42	-6.98	-6.1	-3.75
LOF	Mean	-49.54	-47.06	-40.03	-29.4	-17.2
HST	LOCF	-48.8	-39.43	-32.53	-25.84	-20.82
HST	Interpolation	-51.24	-41.15	-34.23	-29.66	-24.77
IForest	LOCF	-21.19	-28.4	-33.39	-39.52	-48.02
HST	Mean	-79.58	-74.72	-65.08	-56.65	-48.48
IForest	Interpolation	-27.31	-33.82	-39.16	-46.3	-54.99
IForest	Mean	-100.0	-100.0	-100.0	-100.0	-100.0

Experiment No.4 - AirQuality Dataset

Outlier	Imputation	50%	60%	70%	80%	90%
No Action	No Action	-3.56	-3.1	-2.43	-1.8	-1.17
HST	Drop	25.82	22.9	19.39	15.38	12.09
IForest	Drop	0.62	0.75	2.35	4.81	5.22
LOF	Drop	-1.17	-1.27	-1.17	-0.42	0.06
Z	Drop	-3.56	-3.03	-2.44	-1.81	-1.07
Z	Mean	-3.54	-2.99	-2.48	-1.76	-1.13
Z	LOCF	-3.54	-3.0	-2.48	-1.75	-1.14
Z	Interpolation	-3.54	-3.0	-2.48	-1.75	-1.15
LOF	Interpolation	-5.76	-5.07	-4.13	-3.03	-1.73
LOF	LOCF	-6.16	-5.21	-4.34	-3.55	-2.02
HST	Mean	-10.69	-9.56	-7.83	-5.87	-3.39
LOF	Mean	-7.9	-6.97	-6.4	-5.53	-3.72
HST	Interpolation	-10.6	-9.46	-8.38	-6.96	-5.47
HST	LOCF	-10.7	-9.91	-9.22	-7.88	-6.57
IForest	Interpolation	-7.66	-7.09	-6.96	-7.03	-7.4
IForest	LOCF	-8.23	-7.56	-7.2	-7.31	-7.71
IForest	Mean	-12.47	-11.94	-12.1	-12.51	-12.67

Experiment No.4 - NEWeather Dataset

Outlier	Correction	Imputation	50%	60%	70%	80%	90%
No Action	No Action	No Action	-3.56	-3.11	-2.14	-1.63	-0.84
LOF	Drop	Interpolation	0.68	1.15	1.25	1.47	1.01
LOF	Drop	LOCF	1.41	1.61	1.39	1.53	0.88
HST	Drop	Interpolation	-0.76	-0.58	0.17	-0.17	0.58
HST	Drop	LOCF	-0.74	-0.5	0.21	-0.06	0.55
Z	Drop	LOCF	-1.7	-0.88	-0.46	-0.59	0.2
Z	Drop	Interpolation	-1.81	-0.97	-0.45	-0.62	0.2
Z	LOCF	LOCF	-2.7	-2.23	-1.38	-1.2	-0.75
Z	LOCF	Interpolation	-2.83	-2.28	-1.36	-1.27	-0.78
LOF	LOCF	LOCF	-4.46	-3.45	-3.09	-2.3	-1.88
HST	LOCF	LOCF	-2.71	-2.64	-1.66	-1.76	-1.89
HST	LOCF	Interpolation	-2.78	-2.72	-1.64	-1.84	-1.94
LOF	LOCF	Interpolation	-4.4	-3.47	-3.12	-2.22	-1.94

Experiment No.5 - AirQuality Dataset

Outlier	Correction	Imputation	50%	60%	70%	80%	90%
No Action	No Action	No action	-3.11	-2.48	-2.05	-1.43	-0.98
LOF	Drop	LOCF	-1.7	-1.03	-0.61	-0.15	-0.06
LOF	Drop	Interpolation	-1.44	-1.06	-0.63	-0.2	-0.11
Z	Drop	Interpolation	-2.36	-1.88	-1.48	-0.93	-0.62
Z	LOCF	LOCF	-2.54	-2.05	-1.67	-1.07	-0.68
Z	Drop	LOCF	-2.58	-2.04	-1.61	-1.06	-0.71
Z	LOCF	Interpolation	-2.42	-2.05	-1.44	-0.99	-0.72
HST	Drop	LOCF	-2.36	-1.95	-1.57	-1.1	-0.79
HST	Drop	Interpolation	-2.27	-1.91	-1.37	-1.05	-0.81
HST	LOCF	Interpolation	-2.87	-2.15	-1.67	-1.15	-0.98
HST	LOCF	LOCF	-2.84	-2.28	-1.72	-1.27	-1.03
LOF	LOCF	Interpolation	-4.81	-4.06	-3.25	-2.1	-1.34
LOF	LOCF	LOCF	-4.8	-4.29	-3.34	-2.43	-1.4

Experiment No.5 - NEWeather Dataset

Outlier	Correction	Imputation	50%	60%	70%	80%	90%
No Action	No Action	No Action	-3.56	-3.11	-2.14	-1.63	-0.84
LOF	Drop	LOCF	0.95	1.93	1.82	1.58	1.2
LOF	Drop	Interpolation	1.26	1.49	1.58	1.36	1.18
HST	Drop	Interpolation	-0.87	-0.72	-0.03	-0.28	0.53
HST	Drop	LOCF	-0.8	-0.59	0.02	-0.18	0.5
Z	Drop	Interpolation	-1.85	-0.94	-0.62	-0.71	0.08
Z	Drop	LOCF	-1.78	-0.88	-0.57	-0.64	0.07
Z	LOCF	LOCF	-2.72	-2.26	-1.4	-1.21	-0.76
Z	LOCF	Interpolation	-2.87	-2.33	-1.39	-1.31	-0.78
LOF	LOCF	LOCF	-4.34	-3.64	-3.26	-2.35	-1.84
HST	LOCF	LOCF	-2.74	-2.66	-1.69	-1.77	-1.89
HST	LOCF	Interpolation	-2.88	-2.77	-1.69	-1.86	-1.93
LOF	LOCF	Interpolation	-4.61	-3.58	-3.04	-2.14	-1.97

Experiment No.6 - AirQuality Dataset

Outlier	Correction	Imputation	50%	60%	70%	80%	90%
No Action	No Action	No Action	-3.11	-2.48	-2.05	-1.43	-0.98
LOF	Drop	LOCF	-1.35	-1.12	-0.62	-0.31	-0.08
LOF	Drop	Interpolation	-1.15	-1.05	-0.44	-0.19	-0.12
Z	Drop	Interpolation	-2.37	-1.96	-1.48	-0.96	-0.61
Z	LOCF	LOCF	-2.55	-2.08	-1.68	-1.07	-0.68
Z	Drop	LOCF	-2.58	-2.03	-1.59	-1.06	-0.69
Z	LOCF	Interpolation	-2.43	-1.91	-1.49	-1.02	-0.73
HST	Drop	LOCF	-2.5	-2.03	-1.72	-1.13	-0.9
HST	Drop	Interpolation	-2.44	-1.86	-1.63	-1.24	-0.9
HST	LOCF	Interpolation	-2.79	-2.07	-1.55	-1.16	-0.98
HST	LOCF	LOCF	-2.86	-2.31	-1.73	-1.27	-1.03
LOF	LOCF	Interpolation	-4.38	-3.9	-3.09	-2.2	-1.28
LOF	LOCF	LOCF	-4.66	-3.95	-3.13	-2.28	-1.34

Experiment No.6 - AirQuality Dataset

List of Figures

List of Tables

List of Symbols

Symbol	Description
$\boldsymbol{u}$	User
$\boldsymbol{up}$	User Preferences
$\boldsymbol{dpf}$	Data Profiling Features
$\boldsymbol{d}$	Dataset
$\boldsymbol{d}_i$	Contaminated Dataset
$\boldsymbol{mx}_i$	Injection Mask
$\boldsymbol{dpa}_j$	Data Preparation Action
$\boldsymbol{dqd}$	Data Quality Dimension
$\boldsymbol{m}_{ij}$	Performance Metric
$\boldsymbol{w}$	Window
$\boldsymbol{x}$	Window Size
$\boldsymbol{y}$	Window Sliding factor
$\boldsymbol{od}_j$	Outlier Detection Method
$\boldsymbol{o}_{ij}$	Detected Outliers

Acknowledgements

Ringrazio la Professoressa Cappiello per la disponibilità e per avermi seguito durante il lavoro sulla tesi.

Grazie Camilla per avermi aiutato in questi mesi e per la pazienza avuta nel periodo della stesura.

Grazie ai miei genitori per aver reso possibile questo percorso.

Grazie ai miei amici e alla mia famiglia per il supporto che ho ricevuto.