Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 10, No. 03, September 2024.

A MACHINE LEARNING BASED DECISION SUPPORT FRAMEWORK FOR BIG DATA PIPELINE MODELING AND DESIGN

Asma Dhaouadi[1], Khadija Bousselmi[2], Sébastien Monnet[3], Mohamed Mohsen Gammoudi[4] and Slimane Hammoudi[5]

(Received: 25-Mar.-2024, Revised: 13-May-2024, Accepted: 27-May-2024)

ABSTRACT

The data warehousing process requires an architectural revolution to settle big-data challenges and address new data sources, such as social networks, recommendation systems, smart cities and the web to extract value from shared data. In this respect, the pipeline-modeling community for the acquisition, storage and processing of data for analysis purposes is enacting a wide range of technological solutions that present significant challenges and difficulties. More specifically, the choice of the most appropriate tool for the user's specific business needs and the interoperability between the different tools have become primary challenges. From this perspective, we propose in this paper a new interactive framework based on machine learning (ML) techniques to assist experts in the process of modeling a customized pipeline for data warehousing. More precisely, we elaborate first (i) an analysis of the experts' requirements and the characteristics of the data to be processed, then (ii) we propose the most appropriate architecture to their requirements from a multitude of specific architectures instantiated from a generic one, by using (iii) several ML methods to predict the most suitable tool for each phase and task within the architecture. Additionally, our framework is validated through two real-world use cases and user feedback.

KEYWORDS

Big data, Data-warehousing modeling, Modeling assistance, Tools and technologies, ML methods.

1. INTRODUCTION

The technological revolution, the emergence of new Internet services, the blooming growth of smart devices and sensors, mobile and web applications and social media (Facebook, Twitter, Instagram, …etc.) generate a large amount of data daily, known as "Big Data". Notably, Big Data is facing several challenges labeled Vs, like: (i) the Volume presenting the massive amount of data collected by a company, (ii) the Variety which refers to the heterogeneity of data, including structured, semi-structured or unstructured types and (iii) the Velocity, which refers to the speed by which data is collected and needs to be taken into account for eventual processing and decision-making. To address these big-data challenges, numerous platforms, software systems and architectural frameworks have been developed for data warehousing and analysis. However, the diverse landscape of available solutions introduces additional challenges, including the deployment requirement, which addresses verifying the interoperability between the tools, their performance and the experts' technical constraints, such as the resources provided at the deployment phase of the architecture. Consequently, experts need help to select the most suitable tools from a wide range of options. Furthermore, modeling big-data pipelines is crucial before deployment and certain tools prioritize addressing some big-data challenges over others. For example, Apache Kafka does not focus on the veracity of data but rather on its transfer and, thus, on Volume and Velocity [1], while column-oriented tools dedicated to data storage, such as HBase, MongoDB and Cassandra, specifically favor data Variety by supporting different formats and data types [5]. Additionally, the need for adaptability and evolution of data-warehousing and analytics systems is pressing and currently needs to be a standardized architectural solution that guarantees the best selection of tools based on experts' requirements and constraints. To overcome these challenges, we propose in this paper ArchiTectAI: an AI-driven framework to assist experts in selecting the most appropriate tools to meet their particular requirements and constraints

1. A. Dhaouadi is with LISTIC, Savoie Mont Blanc University, France and RIADI, University of Tunis El Manar, Tunisia. Email: asma.dhaouadi74@gmail.com, ORCID 0000-0002-9832-5000.
2. K. Bousselmi is with LISTIC, IUT, Savoie Mont Blanc University, France. Email: khadija.arfaoui@univ-smb.fr
3. S. Monnet is with LISTIC, Polytech, Savoie Mont Blanc University, France. Email: sebastien.monnet@univ- smb.fr
4. M.M. Gammoudi is with RIADI Lab, ISAM La Mannouba, Tunisie. Email: gammoudimomo@gmail.com
5. S. Hammoudi is with ERIS, ESEO-TECH Angers, France. Email: slimane.hammoudi@eseo.fr

when elaborating big-data warehousing and analysis solutions. The main target is to assist the experts' pipeline modeling by considering the specificities of the data to be processed (i.e., the Volume, Velocity, Veracity and Variety) and respecting their preferences. This is achieved by selecting the best tools dedicated to their corresponding needs through the use of personalized ML models employing various ML methods, such as Decision Trees, Random Forest, Support Vector Machine and Gradient Boosting.

In this paper, the main contributions are outlined as follows:

- A hybrid, generic, two-level architecture that supports batch and stream-data processing is proposed to guide the instantiating of architecture models specific to big-data platforms and tools.
- ArchiTectAI: a decision-support framework based on ML that assists experts in modeling big-data pipelines, considering their specific needs and ensuring tool interoperability.
- An *ad-hoc* method for generating a base of tools that considers its alignment with big-data characteristics. This method employs an algorithm that ensures easy maintenance and scalability of the tool base.
- The use of several ML classifiers to predict the most suitable tool for each phase and task of the architecture.

This paper is structured as follows. Section 2 displays the proposed generic big-data pipeline architecture and its phases and tasks. Subsequently, in Section 3, we detail the proposed architecture to support the proposed framework for big-data pipeline modeling. Next, in Section 4, we validate and evaluate the introduced proposals. Section 5 provides a summary of the most prominent related works. Eventually, in Section 6, we conclude the whole work and offer new perspectives for future research.

2. TOWARDS A GENERIC BIG DATA PIPELINE ARCHITECTURE

In this section, we propose a generic architecture for end-to-end big-data warehousing and analysis. The specificities of this architecture are that: (i) it is a generic and Tools Independent Architecture (TIA) that supports two different scenarios. The first one is only for data collected using a single acquisition mode: batch or stream. The second one is the most complete hybrid scenario, which supports both batch and stream-data acquisition modes. As shown in Figure 1, the TIA generic architecture consists of the following four phases:

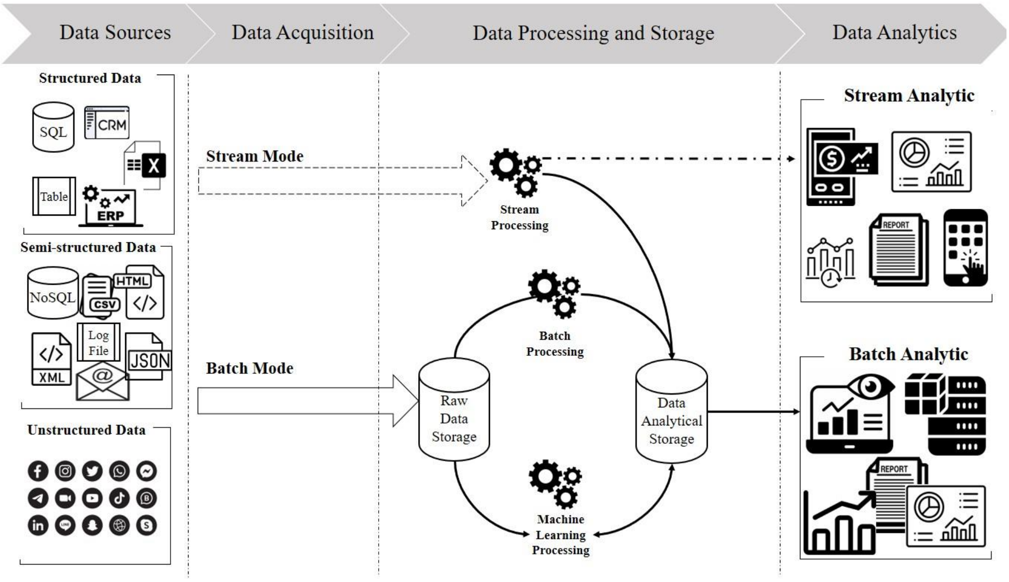


Figure 1. The generic architecture: tool independent architecture.

1) Data Sources phase: This layer gathers the different types of input data, which are classified into structured, semi-structured and unstructured data. The structured data is provided by the tabular data and traditional systems, like data warehouses, data mart, Customer Relationship Management (CRM) or Enterprise Resource Planning (ERP) systems. The semi-structured

data constitutes the NoSQL databases, HTML, JSON, XML and CSV files and e-mails. The unstructured data are generated from social networks, images, audio, videos and data streams, including chat messages shared on the Internet from WhatsApp and mobile SMS text and many other data types like geo-localization data and sensor data from smart devices [2].

2) Data Acquisition phase: Our architecture allows treating three different scenarios at this stage. The first scenario occurs when the data to be acquired, processed and stored is streaming data that necessitates real-time processing and must be ingested instantaneously as it is generated. This scenario is well-suited for mission-critical applications, such as cyber-security monitoring, financial-transaction processing and early-warning systems for natural disasters. The second scenario is when the data to be handled is ingested over a certain period and then processed all at once; this is known as the batch mode. Unlike real-time processing, this deferred processing concerns static data, such as relational tables and Hadoop files [3]. Finally, the third scenario combines streaming and batch data and the architecture will support both data types simultaneously.

3) Data Processing and Storage phase: In our architecture, we merged processing and storage tasks into a single layer to generalize as much as possible and cover stream and batch-processing modes. This middle layer is slightly divided horizontally into two sub-layers: the top layer is specific to stream processing, while the bottom layer corresponds to batch processing. For the batch mode, the data gathered is stored in its raw form and its processing is planned according to the experts' requirements. The processing involves tasks, such as data cleaning, outlier removal and preparation for storage in specific supports dedicated to analysis. Different methods for data cleaning and outlier removal have been outlined in [4]. ML type processing is also proposed at this level to perform increasingly sophisticated treatments in response to the business requirements of experts. In this phase, stream processing, batch processing, ML processing, raw-data storage and data analytic storage constitute the TIA tasks.

4) Data Analytics phase: Data analysis is based on creating dashboards, OLAP navigation, statistics, charts and reports. The stream or batch-data analysis type depends mainly on the data-processing type and data availability in the repositories. It also depends on the level of interactivity that the expert requires to be provided by the dedicated tool. Indeed, the analysis is adapted to experts' needs. Supposing that they require interactive queries, instantaneous processing answers on data collected in real-time or answers on batches of stored data. Each type of analysis is presented by a different icon. For instance, there are forecasting, real-time alerting, dashboards, mobile applications, reporting and visualization for streaming analysis. However, the most frequent business intelligence applications for batch analysis are dashboard, reporting, data visualization and data statistics on which recommendation engines, like Amazon and YouTube videos, can be based.

This generic architecture is designed to be invested for different deployment requirements, meeting the business constraints and the specificities of the experts' needs each time. From this perspective, we can derive different Tools Specific Architectures (TSAs) according to the use case. While, for lack of space in this paper, we will not mention details about the modeling approach formalizing the transition between the TIA level and the TSAs one, we highlight that a modeling approach has been proposed to establish the transition between the generic level and the applicative one by taking full advantage of the Model-Driven Architecture (MDA) approach for software design and development. The modeling details will be provided in a separate paper.

The following section presents the proposed framework for generating different concrete pipelines of TSAs, taking advantage of TIA's genericity.

3. ArchiTectAI: Personalized Big Data Warehousing Advisor

In this section, we present our interactive ML-based framework, assisting the expert in the composition of his/her big-data pipeline and allowing the automatic generation of several TSAs. Figure 2 shows the ArchiTectAI framework's architecture, including three modules: tools database generation, ML-model generation and architecture generation process. In the following part, we detail the role of each module.

3.1 Tools Database Generation

This module is based on an *ad-hoc* method for generating tool databases for big-data warehousing. In these databases, we specified the characteristics of each tool from the categorical variables defining the big data Vs (Volume, Velocity, Veracity and Variety). Among these categorical variables, we note the acquisition mode of data, the data type, the size of data, among others. As detailed in the pseudo-code of Algorithm 1, considering the specific list of tools for each phase and task of TIA and the different categorical variables, we peruse the TIA process and we determine all possible combinations between the different features and the corresponding tools based on predefined rules. The output of this method is a tools database for each TIA phase and task, containing the tools with their corresponding and valuable characteristics for the ongoing step. As depicted in Figure 2, there are distinct databases for data acquisition, raw-data storage, analytical data and data analysis. Regarding data processing, as previously mentioned, there are three types of processing: stream, batch and ML (see Figure 1). In classifying tools into databases, we utilize the criteria of "mode". Tools are categorized based on whether they support stream mode or batch mode, resulting in a single database for both processing modes. Conversely, for ML processing tools, we allocate a separate database, as we consider that ML is a type of treatment supported, rather than a processing mode. In all instances, there are tools performing more than one task, thus being shared by more than one base. For example, Spark, Python and Flume handle ML, batch and stream processing and are therefore present in both the ML data processing and data processing databases.

Algorithm 1 An *Ad-Hoc* Method for Generation of Tools Databases

```
Input 1: Tool Independent Architecture TIA Input 2:
List of Tools ToolsList
Input 3: categoricalVariables catVars
Output: Tools Databases ToolsDBs
TIA ← {TIAphs,TIAtsks}                                   ▷ The TIA phases and tasks T
oolsList ← TLacqui ∪ TLrawStor ∪ TLanalStor ∪ TLprocT ∪ TLprocMLT ∪ TLanal catVars ←
{Acquisition_mode : {“Stream”, “Batch”},
            SizeData: {“over1Peta”,“in1T1P”,“less1Tera”},
            D_type : {“UNS”, “SEMI”, “STRUC”},
            PotentielTimeAcqui: {“RT”,“NRT”,“Batch”},
            Quality: {“LossDup”,“LossData”,“DupData”,“NoLossNoDup”},
            Nb_DS : {“over20”, “in5_20”, “less5”},
            Latency : {“less5”, “in5_15”, “over10”},
            Complexity : {“one”, “multiple”}}
foreach TIAphs AND TIAtsks in TIA do
    foreach T in ToolsList do
        T_Values ← Assign(Val)                  ▷ Assign characteristic values to each tool
        {Val is a value from catVars}
    end
end
foreach T in ToolsList do
▷ Generate specific combinations of each tool by using zip
    foreach t_Values in zip(T_Values) do
        csv_writer.writerow(t_Values + [T])        ▷ Write rows in the “ToolsDBs.csv" file
    end
end
return ToolsDBs
```

This proposed *ad-hoc* method for generating tool bases is conducive to easy updates and maintenance. The process automatically handles all updates by adding the tool with its characteristics expressed in categorical values and developing new data combinations. Then, the generated tool databases will be leveraged as input data by ML methods to predict the most suitable tools while ensuring adherence to the experts' constraints and preferences. As for the categorical variables, they are also used as features in the ML module (sub-section 3.2) and to express user constraints (Tab 2 in subsection 3.3).

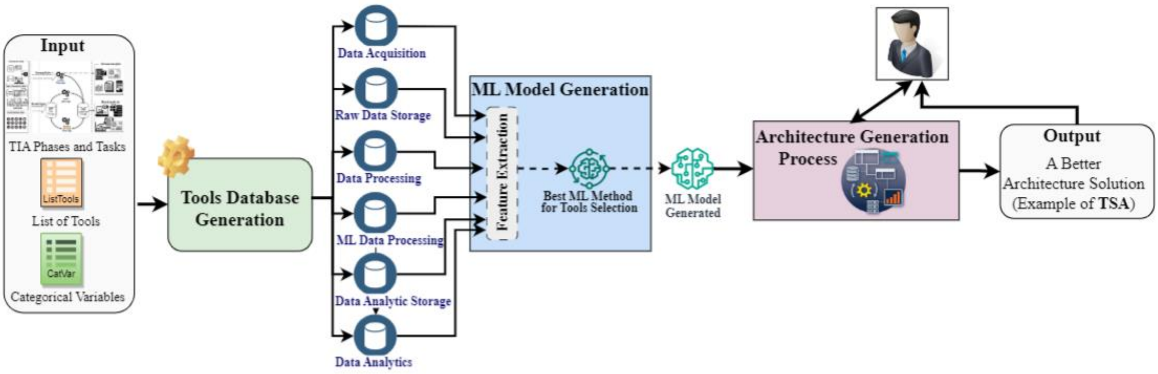


Figure 2. The architecture of ArchiTectAI framework.

3.2 ML Model Generation

This second module of the proposed framework, as shown in Figure 3, is based mainly on two steps: the first is for feature extraction and the second is for ML processing and determining the best ML method. The input data consists of the tool databases specific to each TIA phase and task. Each database contains around 15 tools and their big-data characteristic satisfaction values. Until now, the number of tools handled in the training dataset is 64 and is expected to evolve in line with the technological revolution. As for the result of this module, the generated ML models are to be used in predictions of the best tools for subsequent use by the experts. In the following part, we outline the two steps of this module and provide a comprehensive overview of the experiment conducted to motivate the chosen ML method.

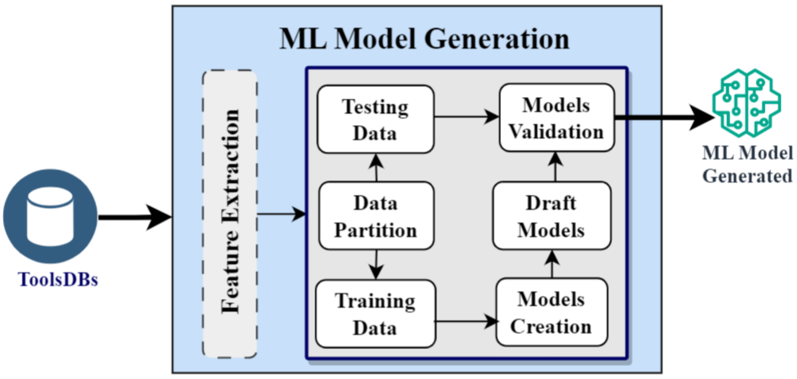


Figure 3. The machine-learning model generation process.

3.2.1 Feature Extraction

Since each phase and task of TIA has specific characteristics and must meet particular constraints, the features expressed as categorical variables in the above tool databases generation module are different for every task and phase of the pipeline. Moreover, each tool database has dedicated tools to perform the processing required for the corresponding task. For these reasons, for each TIA step, we have specified a particular ML processing with the corresponding tool database as input; the features are the tool characteristics and their target variables are the names of the tools.

3.2.2 ML Processing and Determining the Best ML Method

Our framework aims to predict the tool that best meets the experts' constraints and the specific data to be processed by the architecture. Leading to this prediction, we have opted for supervised learning, a branch of ML using algorithms to analyze the relationship between the constraints of each TIA phase and the tasks designated as features (e.g. SizeData, D_type, Latency, …etc.) and the tools stored in the corresponding database. However, in this instance, since each phase has its specific tool database and characteristics and to provide our framework with greater flexibility, we have opted to implement several ML methods and conduct performance tests, assuming that each ML method can perform differently and provide different results. The implemented ML methods in our process consist of Decision Trees, Random Forest, Support Vector Machine and Gradient Boosting.

The performance-evaluation measures examined in this module include Accuracy, Precision, Recall and F1-Score. In our context, we performed the following steps: Initially, we divided the input data into training, validation and testing sets. Then, we compared the performance of the different ML methods based on the specified performance metrics. Next, we trained each method on the training set using the extracted features. We optimized the hyperparameters of each classifier using grid search and cross-validation with a fold value of 5 (cv=5). The performance of each classifier was evaluated on the validation set utilizing the defined performance metrics. Finally, we choose the model generated by the best ML classification method in each phase to be used later in the architecture-generation process module.

As shown in Table 1, the evaluation of the implemented ML methods shows excellent results in the different phases. This asserts the effectiveness and reliability of the models generated, which are extremely useful for the framework’s ability to provide precise, helpful advice on tool selection and pipeline implementation.

Despite the results on most phases (5/6) leading to the deduction that the decision tree has performed well, we have consolidated the evaluation by calculating the average of measures per metric over the whole pipeline. This confirms that the decision tree is the best method deployed to generate ML models and predict the best tool for each phase and task in the pipeline to the expert.

Table 1. Experimental results to determine the best ML method.

ML Method		**Decision Tree**				**Random Forest**				**Support Vector Machine**				**Gradient Boosting**			
Performance Metrics		**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Architecture Tasks	Data Acquisition	**0,83**	0,79	0,84	0,81	0,83	0,78	0,8	0,79	0,81	0,75	0,79	0,77	0,8	0,75	0,83	0,79
	Data Processing	**0,82**	0,79	0,84	0,81	0,8	0,74	0,83	0,78	0,8	0,75	0,79	0,77	0,79	0,76	0,79	0,77
	ML Data Processing	0,8	0,71	0,8	0,75	**0,84**	0,81	0,84	0,82	0,78	0,57	0,71	0,63	0,76	0,61	0,78	0,68
	Raw Data Storage	**0,83**	0,8	0,86	0,82	0,83	0,78	0,82	0,80	0,76	0,66	0,73	0,69	0,78	0,67	0,78	0,72
	Data Analytic Storage	**0,82**	0,76	0,87	0,81	0,78	0,76	0,81	0,78	0,77	0,68	0,78	0,73	0,76	0,69	0,78	0,73
	Data Analytics	**0,81**	0,79	0,88	0,83	0,8	0,74	0,82	0,78	0,79	0,69	0,81	0,75	0,77	0,77	0,85	0,81
Average Measures		**0,82**	0,77	0,85	0,81	0,81	0,77	0,82	0,79	0,79	0,68	0,77	0,72	0,78	0,71	0,80	0,75

Overall, in these experiments, we observed that the limited size of the tool databases slightly impacted the performance of the ML methods. However, the scalability and ease of maintenance and updating of the various modules of the framework allow for improving the performance of ML methods by expanding the number of supported tools. Indeed, the developed prediction models in the different pipeline phases and tasks can be easily updated when new data traces are available or the process model changes. This flexibility stems from the prediction model incorporating independent databases for each phase and task and we have appropriate characteristics for each corresponding tool. Finally, by leveraging ML methods and the relationships identified through supervised learning, we evaluated and validated the best ML method, allowing our proposed framework for effective tool selection at each pipeline phase and task.

In the next sub-section, we will detail the generation process of TSAs that meet the experts’ needs and we will explain how they exploit the generated ML models.

3.3 Architecture Generation Process

The interactivity of our framework is based on multiple real-time exchanges between the expert and the framework during the various phases of the process of generating a tailored model of the TSA. As shown in Figure 4, this process consists of five phases: Predesign phase, Data Acquisition, Data Storage and Processing, Data Analytics and Data Consumption. After the authentication step, in the Predesign phase, the expert expresses the constraints related to his/her application case and the data specificities to be supported by the pipeline, such as data size, data type, acquisition mode and many other details like the intended analytic application, using a form. We have categorized the expert constraints in Table 2 according to the big-data V-challenges and the categorical variables previously used as features for generating ML models. Then, in each phase, the framework first displays all the tools available to perform the current task. Next, it runs the ML model corresponding to this task to predict the most suitable tools for the expert’s needs, gathered from the form and the task’s specificities. Given the variety of tools studied in the ML phase, the prediction result may consist of several proposed tools. In this case, the expert selects the tool according to his/her preferences and validates his/her choice before proceeding to the next task.

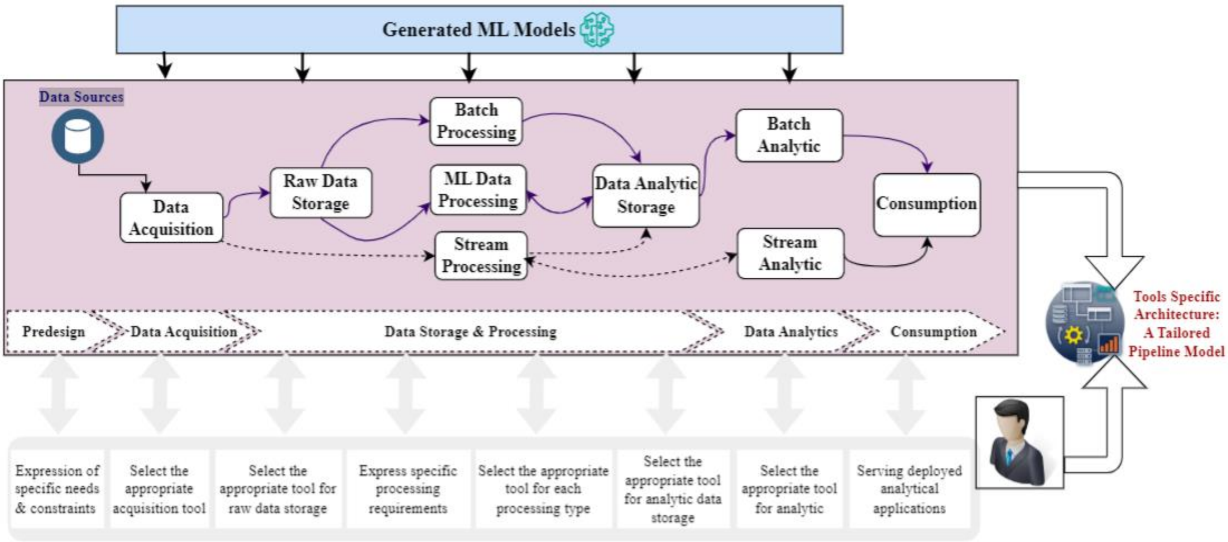

Generated ML Models
Data Sources
Data Acquisition
Raw Data Storage
Batch Processing
ML Data Processing
Stream Processing
Data Analytic Storage
Batch Analytic
Stream Analytic
Consumption
Predesign
Data Acquisition
Data Storage & Processing
Data Analytics
Consumption
Expression of specific needs & constraints
Select the appropriate acquisition tool
Select the appropriate tool for raw data storage
Express specific processing requirements
Select the appropriate tool for each processing type
Select the appropriate tool for analytic data storage
Select the appropriate tool for analytic
Serving deployed analytical applications
Tools Specific Architecture: A Tailored Pipeline Model

Figure 4. Architecture generation process's pipeline.

Moreover, given the importance of interoperability checking between tools of different phases and the subsequent issues arising during the deployment of a big-data architecture, we have conducted a detailed an interoperability study as detailed in [5] to which we refer interested readers. We have leveraged this study's results to deploy our framework. In this way, when the expert selects a tool that is non-interoperable with the tool already chosen in the previous phase, an alert message is displayed and consequently, the expert should choose another tool from the list predicted by the ML model. On the other hand, in some phases of the process, tasks require more details from the expert. For example, in the data processing task, to allow the process to predict the best tool, the framework requests the expert to express its requirements regarding the response time that the proposed tool should meet. The expert's interaction with the framework continues until the data consumption phase, which contains the data results to be consumed by the chosen tool in the data-analytics phase. We then find the analytical applications requested by the expert in the initial form, such as OLAP Navigation, dashboard, reporting, mobile application, forecasting, real-time analytics, statistics, visualization, …etc. Finally, at the end of the process, the expert recovers a tailored pipeline model that meets his/her specific constraints and the requirements of his/her application case, which he/she has been involved in composing step by step; this is its tailor-made end-to-end pipeline model for TSA.

Table 2. Experts' constraints expressed in terms of big-data features.

Big-data Features	**Volume**	**Velocity**	**Veracity**		**Variety**			
Expert's Constraint	**Size of Data**	**Time Acquisition**	**Response Time**	**Quality**	**Number of Data Sources**	**Data Type**		**Complexity**
Categorical Value	over1Peta	Real Time	less5	Loss and duplicate data	over20	Table, CSV, SQL DB, Spreadsheets, Log File	Structured	One model
	in1T1P	Near Real Time	in515	Loss data	in5_20	JSON, XML, Social Media	Semi-structured	Multiple models
	less1Tera	Batch	over15	Duplicate data	less5	Sensor Data, NoSQL DB, Video, Images, Geo-spacial Data	Unstructured	
				No loss nor duplicate				

In summary, our interactive, ML-based framework provides an entirely automatic process. The expert needs to express his/her requirements and is guided step by step to the final phase, where he/she obtains a pipeline model in which all the tools are interoperable. The following section will validate this framework by applying it to two use cases.

4. ArchiTectAI: Validation and Evaluation of Results

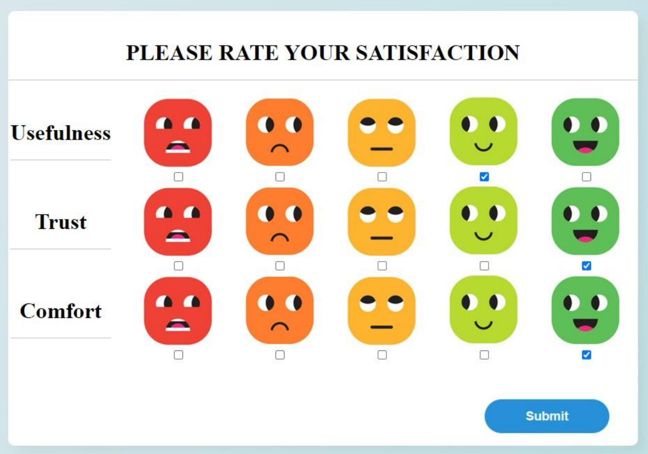
For ArchiTectAI evaluation, we opted for the ISO/IEC 25022 SQuaRE standard[1], in particular the user-satisfaction characteristic [6]. As defined in [7], satisfaction involves three sub-characteristics:

- Usefulness: "The degree of user satisfaction with achieving the objectives, including the results and consequences of use" [7].
- Trust: "The degree to which a user has confidence that a software product will perform as intended" [7].
- Comfort: "The extent of the user's satisfaction with physical comfort" [7].

We have developed a specific module for collecting and analyzing expert feedback to measure these criteria. As shown in Figure 5, we propose five levels of evaluation: Excellent, Good, Average, Poor and Very Poor. Next, we reached out to twenty experts who engage with big data for various purposes,requesting their participation in testing the framework against their respective requirements. Subsequently, we gathered their feedback, obtained upon the completion of the pipeline-generation process. Table 3 presents a comprehensive summary of these findings. Overall, the responses are very satisfactory for the majority and show that the framework has met the specific expectations of the experts.

For ArchiTectAI validation, we defined valuable criteria that were adapted to our context. For each criteria, we proposed the following evaluation questions:

- The consistency of the pre-design phase. Q1: Does the questions asked in the form cover all the experts' requirements and data specificities?
- Expert-framework interaction. Q2: Are the interactions satisfactory from an ergonomic perspective, particularly regarding usability and guidance?
- Clarity of the process for proposing and validating tool choice. Q3: Is it clear how the predicted tools are presented to the experts? Is the task of validating the choices simple?



PLEASE RATE YOUR SATISFACTION

Usefulness	☐	☐	☐	☑	☐
Trust	☐	☐	☐	☐	☑
Comfort	☐	☐	☐	☐	☑

Figure 5. ArchiTectAI evaluation by measuring the experts' satisfaction levels.

Table 3. Satisfaction evaluation.

Satisfaction	Usefulness	Trust	Comfort
Excellent	7	8	9
Good	10	10	9
Average	3	2	2
Poor	0	0	0
Very Poor	0	0	0

- Functional framework. Q4: When I implement the proposed TSA, do I have problems with tool interoperability?
- The expected results. Q5: After implementing the proposed TSA, do the analytical applications identified in the data-consumption phase meet the business needs?
- Technical constraints addressed. Q6: Did the ArchiTectAI consider the technical environment

[1] Systems and software-quality requirements and evaluation

of the experts when proposing the tools?

- Framework evaluation. Q7: Does the ArchiTectAI enable experts to express their satisfaction?

For the validation process, we have applied ArchiTectAI to two use cases, which have been addressed in previous works: [5] and [8]. For each use case, we proceeded as follows: We acted as the expert and followed the steps proposed by ArchiTectAI to generate the corresponding TSA. We checked how it assisted us and we assigned the symbol 'X' if ArchiTectAI validates the criteria expressed by the corresponding question. The results of this evaluation are displayed in Table 4.

4.1 Twitter Data Use Case Validation

In our previous work in [5], an interoperability and experimental study revealing the capabilities of the tools and their resource-consumption requirements were conducted. Two different pipelines were deployed for this experimental study to evaluate the popularity of teams and players before the start of the 2022 World Cup. In order to lead the validation process, we used ArchiTectAI to re-generate the two pipelines (examples of TSA). In the first pipeline, we acquired a dataset of approximately 80 million tweets in JSON format extracted from Twitter's general thread around November 2022, amounting to 500GB of data to be processed for batch-processing purposes. In the second one, we emulated a stream of approximately 1200 tweets per second for stream-processing purposes. In the pre-design phase, we selected in the form the options that addressed the specific characteristics of the data to be processed, e.g. data size, acquisition mode, data type, …etc. (Q1 validated). We then proceeded with all the steps guided by ArchiTectAI in continuous exchange throughout the process. For example, in the processing phase, ArchiTectAI demands the required response time as an additional request specific to the current phase (Q2 validated). At each step, the selection and validation of tool choices are simple and clear. Furthermore, in each phase of ArchiTectAI, a dynamic architecture diagram is generated using the Mermaid[2] technique, which assigns the tool to the current phase (Q3 validated). The previous work aimed to perform statistical reporting to achieve our analytical objective. The ArchiTectAI framework suggested uses Tableau tool, which we had already deployed in our architecture for generating reports; thus, the result was satisfactory (Q5 validated). Moving forward in the process, in the end, as shown in Figure 6, with the assistance of ArchiTectAI, we reproduced the pipeline-architecture model that we deployed in our case study. When implementing the architecture, we did not meet any issues of tool interoperability or a bottleneck for our processing engine. This proves that the proposed pipeline by ArchiTectAI is aligned with our technical constraints (Q4 and Q6 validated).

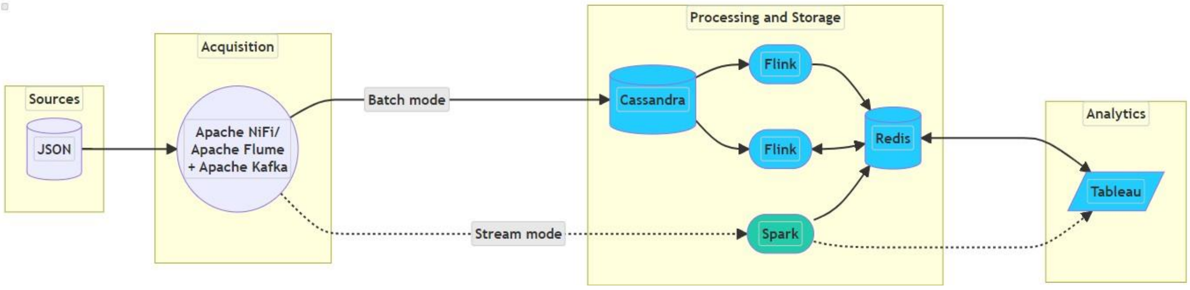


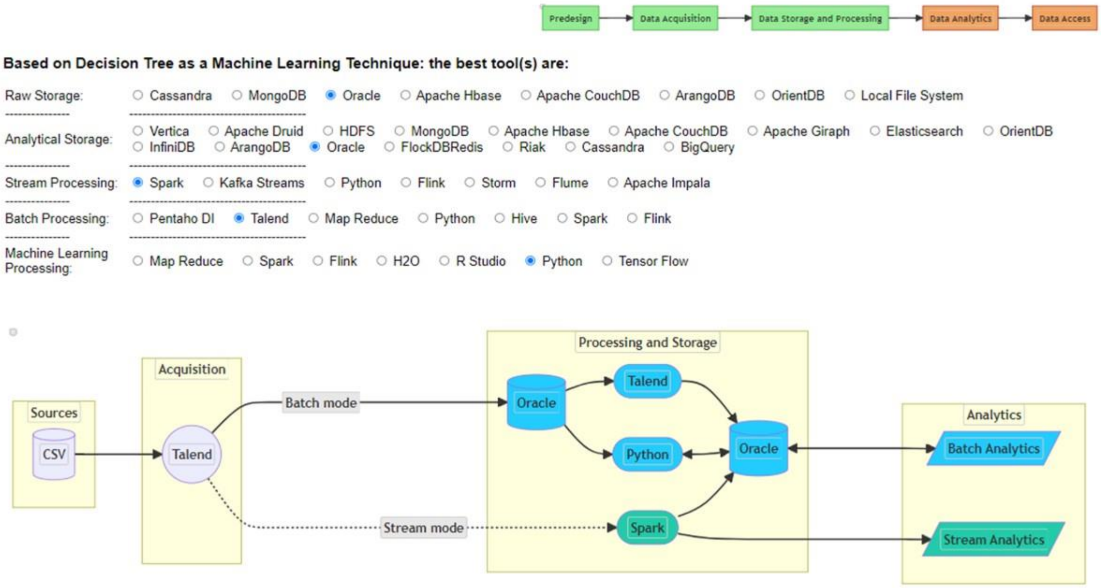
Figure 6. Tools specific architecture model generated for the Twitter data case study.

4.2 The Covid Pandemic Use Case Validation

In our previous work in [8], we introduced a multi-layer model for generating architectures for big-data warehousing. In this research study, we implemented an architecture for storing and analyzing multi-source data to examine the impact of the COVID-19 pandemic evolution on Twitter. This hybrid architecture supported streaming data from Twitter and batch data corresponding to the statistics collected on vaccination campaigns. To validate ArchiTectAI, we applied all its phases, initially, by specifying in the form all the business requirements and data specificities defined in this case study. Then, the responses to this form were analyzed and processed by ArchiTectAI in order to be available for the ML prediction model. As shown in Figure 7, particularly in the data-storage and processing phase, ArchiTectAI proposed a set of tools, in which we found those already deployed in the previous

[2] https://mermaid.js.org/

case study architecture [8]. Thus, we selected them, validated the choices and proceeded to the next phase (Q1 validated). Even for analysis tools, ArchiTectAI suggested tools already in previous use for dashboard creation, reporting and statistics to track the pandemic and vaccination campaigns. So, the expected result of the analysis would also be the same (Q5 validated). Regarding questions Q2 and Q3, we have encountered no problems. In fact, ArchiTechtAI's tool recommendations were clear, the choice was simple and the navigation to move from one phase to another was seamless. Moreover, during the implementation of our architecture, we had no interoperability problems. However, using Excel with a large amount of data caused a bottleneck (Q4 and Q6 validated). Finally, for both case studies, we completed the satisfaction survey proposed at the end of the ArchiTectAI process (Q7 validated).



Based on Decision Tree as a Machine Learning Technique: the best tool(s) are:

Raw Storage: ○ Cassandra ○ MongoDB ● Oracle ○ Apache Hbase ○ Apache CouchDB ○ ArangoDB ○ OrientDB ○ Local File System

Analytical Storage: ○ Vertica ○ Apache Druid ○ HDFS ○ MongoDB ○ Apache Hbase ○ Apache CouchDB ○ Apache Giraph ○ Elasticsearch ○ OrientDB ○ InfiniDB ○ ArangoDB ● Oracle ○ FlockDBRedis ○ Riak ○ Cassandra ○ BigQuery

Stream Processing: ● Spark ○ Kafka Streams ○ Python ○ Flink ○ Storm ○ Flume ○ Apache Impala

Batch Processing: ○ Pentaho DI ● Talend ○ Map Reduce ○ Python ○ Hive ○ Spark ○ Flink

Machine Learning Processing: ○ Map Reduce ○ Spark ○ Flink ○ H2O ○ R Studio ● Python ○ Tensor Flow

Figure 7. Tools specific architecture model generated for the Covid-19 pandemic case study: An overview of tool selection.

In summary, our ArchiTectAI interactive framework provided TSAs implemented in both use cases, proving that they are functional, consistent and support all data specifics and required constraints. They also proved that they met the professional requirements of both studies, with careful analysis of these requirements and supported by ML methods to predict the corresponding tools. Therefore, we have successfully validated ArchiTectAI, a decision-support framework, for big-data pipeline modeling, with particular emphasis on these two specific use cases on which we have extensively worked. However, the applicability of our framework extends far beyond these scenarios. Indeed, our overarching goal was to develop a highly generic framework that can be tailored to a diverse array of big-data application domains and use cases. For example, this includes processing streaming videos in real-time (e.g. ground traffic control), images (medical, satellite, …etc.), textual data (e.g., analyzing sentiment on social media) and other similar applications.

Table 4. Evaluation of validation questions by application of use cases.

Use Case	**Validation Question**						
	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Twitter Data	X	X	X	X	X	X	X
Covid Pandemic Data	X	X	X	X	X	X	X

5. Related Works and Ascertainments

This section is structured as follows. Sub-section 5.1 focuses on approaches addressing big-data challenges, while other approaches examine and evaluate big-data tools. In sub-section 5.2, we review a selection of research work that classifies some big-data tools within proposed architectures, with the purpose of facilitating a choice between them.

5.1 Big-data Warehousing and Analysis: Tools, Technologies and Big-data Features

Day to day, big-data features are raising new challenges for data-warehousing systems. Research studies have been conducted in the literature tackling particular challenges [9][10][11]. In [9], the authors focused on the variety issue by proposing an architectural design of a schema-less big-data repository aiming at capturing all data types. To cope with velocity, the authors in [10] elaborated an approach for detecting concept drift by investing in ML techniques utilizing both real and artificial data. As for Yousfi et al., they proposed in [11] a framework combining different processing engines in order to handle velocity. These engines operate parallel to perform the relevant matching and deliver the most complete and accurate data insight. On the other side, with the emergence of a wide variety of big-data tools, we notice that many surveys have been conducted to discuss and even evaluate these tools [12]-[13], [18]-[19]. However, none of these comparisons have classified the tools according to their satisfaction with big-data features. In particular, in [12], the authors compared some popular big-data frameworks based on the employed-programming model, the types of data sources utilized, the programming languages supported, the fault-tolerance support, the scalability and whether or not iterative processing is supported. Additionally, in [13], the authors considered scalability, distributed architecture, parallel computation and fault tolerance as comparison criteria. Recently, in [20], the author addressed a theoretical study of the big-data value chain, without focusing on specific technological solutions or practical implementations. It explored the conceptual relationships between big-data characteristics and the different stages of the value chain, but did not provide operational details on implementation and interoperability constraints between technologies.

5.2 Big-data Warehousing and Analysis: Approaches to Tool Selection

In [14], the authors defined criteria for choosing big-data tools and proposed a customer data analytics architecture. Despite the originality of their work, the approach has been limited to a restricted selection of tools, focusing only on smart-grid application. In [15], the authors proposed an approach utilizing key performance indicators (KPI), weightage and scores to help choose the best-ranked data warehousing tool for enterprises. In [16], the authors set forward a big-data analytical approach architecture. They classified the investigated approaches for analytical processing into NoSQL-based architecture, parallel relational database-based architecture and graph-based architecture. For each type of these architectures, they examined a set of tools according to these criteria: query language used, scalability, OLAP support, fault-tolerance support, cloud support, programming model and ML support. Moreover, the work in [17] is relevant to the scope of our research. The authors aimed to incorporate an iterative methodology for defining big-data analytics architectures. With its various phases, this methodology covers all the modeling tasks that a designer should perform to define a big-data pipeline. By considering the phase requirements regarding big-data characteristics, the authors introduced some technologies that can be deployed to meet these needs. Despite the importance of the proposed methodology, we note that they did not propose an automatic and interactive solution to guide the users in their choice of tools for each phase of the pipeline. In [20], by examining 110 significant and recently published articles, the authors conducted a comprehensive and systematic literature review on big-data management (BDM) techniques in the Internet of Things (IoT). They categorized the investigated mechanisms into four groups: BDM processes, BDM architectures/frameworks, quality attributes and types of big-data analysis. A detailed comparative analysis was provided for each category. Moreover, the authors presented a holistic BDM framework for IoT, including the following steps: data collection, communication, data ingestion, storage, processing and analysis and post-processing. The reviewed articles were classified according to these framework steps. Additionally, the study evaluated and compared various tools, platforms and frameworks used in the IoT domain based on qualitative criteria, such as performance, efficiency, accuracy and scalability. Finally, despite the comprehensive study presented in this paper and the advice derived from the authors' and other researchers' experiences, it's important to note that it exclusively focuses on techniques deployed in IoT.

Despite the community's awareness of the technological revolution associated with big data and the numerous efforts enacted to compare tools and propose approaches for designing big-data pipelines, we note the following shortages: 1- None of these works has proposed a generic architecture from which we can instantiate multiple specific pipeline models dedicated to different use cases. 2- The proposed approaches are limited to specific case studies. 3- The proposed approaches handle a limited number of big-data tools and often do not focus on checking the interoperability between the proposed tools and the overall consistency of the proposed pipeline. 4- None of the studies proposed a method for creating a tool database classified according to satisfaction with big-data characteristics. 5- None of these works erected an automatic framework based on interaction with the experts to deduce from an exchange form all the particular needs, data specificities and technical constraints. 6- None of the proposed solutions relies on an ML model to analyze the experts' specific needs and identify the most appropriate tools. 7- None of the suggested approaches provide step-by-step assistance to experts composing their end-to-end big-data pipeline model. To address all these issues, we have proposed this ML-based interactive framework driven by a generic architecture to assist experts step-by-step in designing a big-data pipeline customized to their specific needs.

6. Conclusion

This research paper proposes an architecture to support a big data pipeline modeling interactive framework based on ML (ArchiTectAI). This architecture is based on three main modules. An *ad-hoc* method for generating tool databases has been developed for the first module. This method, from the list of tools for each Tools Independent Architecture phase and task and the different categorical variables characterizing big-data challenges, generates tool bases categorized according to their characteristics for each task in the big-data pipeline. The second module generates ML models. This module has implemented and evaluated several ML methods to choose the best one. The third module relies on these ML models to predict the best tool for each task and pipeline phase for the experts while respecting the constraints and specificities of the data. At the completion of this research, we evaluated the satisfaction of our interactive framework based on the ISO/IEC 25022 standard and validated it on two use cases. The consistency of the resulting pipeline proves the framework's effectiveness in its choice for suggesting tool choices. As a final note, this research work is extremely valuable and promising, as it opens further fruitful lines of investigation and offers promising future research directions. Indeed, our framework has been developed to be generic and scalable. Its adaptability allows for future updates with changes to existing tools or the addition of new ones, facilitated by the flexible underlying method for generating the tool database on which it depends. Furthermore, it can also be enriched with additional forms to address more constraints and expert-specific requirements. We also intend to enrich the framework with comprehensive guidelines for deploying the proposed architecture, specifying the connectors between tools. In addition, the tools and platforms for big-data governance and security are beyond this research's scope. In this respect, the elaborated work can be extended by incorporating this type of tools.

References

[1] T. P. Raptis and A. Passarella, "A Survey on Networked Data Streaming with Apache Kafka," IEEE Access, vol. 11, pp. 85333-85350, 2023.

[2] S. Mishra and A. Misra, "Structured and Unstructured Big Data Analytics," Proc. of the 2017 IEEE Int. Conf. on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), pp. 740-746, Mysore, India, 2017.

[3] A. Davoudian and M. Liu, "Big Data Systems: A Software Engineering Perspective," ACM Computing Surveys (CSUR), vol. 53, no. 5, pp. 1-39, 2020.

[4] K. Rahul, R. K. Banyal and N. Arora, "A Systematic Review on Big Data Applications and Scope for Industrial Processing and Healthcare Sectors," Journal of Big Data, vol. 10, Article no. 133, 2023.

[5] A. Dhaouadi, W. Paccoud, K. Bousselmi, S. Monnet, M. M. Gammoudi and S. Hammoudi, "Big Data Tools: Interoperability Study and Performance Testing," Proc. of the IEEE Int. Conf. on Big Data, MIDP Workshop (MIDP-2023), pp. 2386-2395, 2023.

[6] ISO/IEC, "ISO/IEC 25022:2016 - Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Measurement of Quality in Use," ISO/IEC 25022:2016, [Online], Available: https://www.iso.org/standard/35746.html, 2016.

[7] J. Sulla-Torres, A. Gutierrez-Quintanilla, H. Pinto-Rodriguez, R. Gómez-Campos and M. A. Cossio-Bolaños, "Quality in Use of an Android-based Mobile Application for Calculation of Bone Mineral Density with the Standard ISO/IEC 25022," IJACSA, DOI: 10.14569/IJACSA.2020.0110821, 2020.

[8] A. Dhaouadi, K. Bousselmi, S. Monnet, M. M. Gammoudi and S. Hammoudi, "A Multi-layer Modeling for the Generation of New Architectures for Big Data Warehousing," Proc. of the 36th Int. Conf. on Advanced Information Networking and Applications (AINA- 2022), vol. 2, pp. 204–218, 2022.

[9] A. M. Olawoyin, C. K. Leung, C. CJ. Hryhoruk and A. Cuzzocrea, "Big Data Management for Machine

Learning from Big Data," Proc. of the 37th Int. Conf. on Advanced Information Networking and Applications (AINA-2023), vol. 1, pp. 393–405, 2023.

[10] A. Abbasi, A. R. Javed, C. Chakraborty, J. Nebhen, W. Zehra and Z. Jalil, "ElStream: An Ensemble Learning Approach for Concept Drift Detection in Dynamic Social Big Data Stream Learning," IEEE Access, vol. 9, pp. 66408–66419, 2021.

[11] S. Yousfi, D. Chiadmi and M. Rhanoui, "Smart Big Data Framework for Insight Discovery," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 10, pp. 9777–9792, 2022.

[12] W. Inoubli, S. Aridhi, H. Mezni, M. Maddouri and E. M. Nguifo, "An Experimental Survey on Big Data Frameworks," Future Generation Computer Systems, vol. 86, pp. 546–564, 2018.

[13] S. Riaz, M. U. Ashraf and A. Siddiq, "A Comparative Study of Big Data Tools and Deployment Platforms," Proc. of the IEEE Int. Conf. on Engineering and Emerging Technologies (ICEET), pp. 1–6, Lahore, Pakistan, 2020.

[14] H. Daki, A. El Hannani, A. Aqqal, A. Haidine and A. Dahbi, "Big Data Management in Smart Grid: Concepts, Requirements and Implementation," Journal of Big Data, vol. 4, no. 1, pp. 1–19, 2017.

[15] M. R. Sureddy and P. Yallamula, "Approach to Help Choose Right Data Warehousing Tool for an Enterprise", Int. J. of Advance Research, Ideas and Innovat. in Technol., vol. 6, no. 4, pp. 579-583, 2020.

[16] Y. Cardinale, S. Guehis and M. Rukoz, "Classifying Big Data Analytic Approaches: A Generic Architecture," Proc. of the 12th Int. Joint Conf. on Software Technologies (ICSOFT), Part of the Book Series: Communi. in Computer and Information Science, vol. 868, pp. 268-295, Madrid, Spain, 2018.

[17] R. Tardio, A. Mate and J. Trujillo, "An Iterative Methodology for Defining Big Data Analytics Architectures," IEEE Access, vol. 8, pp. 210597–210616, 2020.

[18] S. Alkatheri, S. A. Abbas and M. A. Siddiqui, "A Comparative Study of Big Data Frameworks," Int. J. of Computer Science and Information Security (IJCSIS), vol. 17, no. 1, pp. 66-73, 2019.

[19] M. Khalid and M. Murtaza Yousaf, "A Comparative Analysis of Big Data Frameworks: An Adoption Perspective," Applied Sciences, vol. 11, no. 22, p. 11033, 2021.

[20] A. A. Aydin, "A Comparative Perspective on Technologies of Big Data Value Chain," IEEE Access, vol. 11, pp. 112133 – 112146, 2023.

[21] A. Naghib, N. J. Navimipour, M. Hosseinzadeh and A. Sharifi, "A Comprehensive and Systematic Literature Review on the Big Data Management Techniques in the Internet of Things," Wireless Networks, vol. 29, no. 3, pp. 1085-1144, 2023.

ملخص البحث:

إنّ عملية استيداع البيانات تتطلّب ثورةً بِنيويةً لتسوية تحدّيات البيانات الضّخمة والتّعامل مع مصادر البيانات الجديدة، مثل شبكات التّواصُل الاجتماعي، وأنظمة التّوصية، والمدن الذّكية، وشبكة الويب لاستخلاص قيمةٍ من البيانات المتبادَلة. وفي هذا الإطار، فإنّ مجتمع نمذجة خطوط البيانات من أجل اكتساب البيانات وتخزينها ومعالجتها بهدف تحليلها يوظّف طيفاً واسعاً من الحلول التّكنولوجية الّتي تنطوي بدورها على تحدّياتٍ مهمّة وصعوباتٍ جمّة. وبشكلٍ أكثر تحديداً، فإنّ اختيار الأداة المثلى الملائمة لاحتياجات العمل الخاصّة بالمستخدم وموضوع تبادُلية التّشغيل بين الأدوات المختلفة أصبح من بين التّحديات الأساسية.

من هذا المنطلق، نقترح في هذه الدراسة إطار عملٍ تفاعلياً جديداً مبنياً على تقنيات تعلُّم الآلة لمساعدة الخبراء في نمذجة خطّ بياناتٍ من أجل استيداع البيانات. وعلى نحوٍ أدّق، فإننا نعمل على:

أ) تحليل متطلّبات الخبراء وخصائص البيانات المطلوب معالجتها.
ب) اقتراح البِنية الملائمة لتلك المتطلّبات من بين مجموعةٍ من البِنى.
ج) تحقيق ذلك من خلال عددٍ من طرق تعلّم الآلة لتوقُّع أكثر الأدوات المناسبة لكلّ مرحلة ولكلّ مهمّة داخل البِنية.

بالإضافة إلى ذلك، جرى التّحقُّق من فاعلية إطار العمل المقترح باستخدام حالتَي استخدامٍ من العالم الحقيقي، إلى جانب التّغذية الرّاجعة من المستخدمين.