Received 19 February 2023, accepted 16 April 2023, date of publication 2 May 2023, date of current version 16 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3272503

RESEARCH ARTICLE

Ask Your Data–Supporting Data Science Processes by Combining AutoML and Conversational Interfaces

SARA PIDÓ[1], PIETRO PINOLI[1], PIETRO CROVARI[1], FRANCESCA IEVA[2,3], FRANCA GARZOTTO[1], AND STEFANO CERI[1], (Member, IEEE)

[1]Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133 Milan, Italy
[2]Department of Mathematics, Politecnico di Milano, 20133 Milan, Italy
[3]Center for Health Data Science, Human Technopole, 20157 Milan, Italy

Corresponding author: Pietro Pinoli (Pietro.Pinoli@polimi.it)

ABSTRACT Data Science is increasingly applied for solving real-life problems, in industry and in academic research, but mastering Data Science requires an interdisciplinary education that is still scarce on the market. Thus, there is a growing need for user-friendly tools that allow domain experts to directly apply data analysis methods to their datasets, without involving a Data Science expert. In this scenario, we present DSBot, an assistant that can analyze the user data and produce answers by mastering several Data Science techniques. DSBot understands the research question with the help of conversation interaction, produces a data science pipeline and automatically executes the pipeline in order to generate analysis. The strength of DSBot lies in the design of a rich domain specific language for modeling data analysis pipelines, the use of a suitable neural network for machine translation of research questions, the availability of a vast dictionary of pipelines for matching the translation output, and the use of natural language technology provided by a conversational agent. We empirically evaluated the translation capabilities and the autoML performances of the system. In the translation task, it obtains a BLEU score of 0.8. In prediction tasks, DSBot outperforms TPOT, an autoML tool, in 19 datasets out of 30.

INDEX TERMS Automated machine learning, data science, human–computer interaction, intelligent systems, natural language understanding, pipeline optimization, python.

I. INTRODUCTION

Data Science, situated at the intersection between computer science and statistics, has recently emerged as a new discipline, providing methods that are suitable to explore research questions for arbitrary application domains. In the industrial setting, companies exploit data to optimize processes, predict revenues, and prevent failures by planning predictive maintenance interventions [1]. In research, scientists use data to validate or formulate novel hypotheses and support decision-making [2]. The process of data democratization amplifies these advantages: more and more data repositories are published online to be freely used by researchers all over the world [3].

Still, leveraging the increasing availability of data requires advanced capability in data management and modeling, statistics, machine learning, and programming. As a result, Data Science so far is not fully accessible for domain experts, who may lack strong technical skills and computational background. Data Science tasks are often challenging also for both novice and experienced data scientists. Every dataset is different and requires a tailored sequence of operations to extract useful knowledge [4]. Researchers may adopt a non-optimal analysis pipeline or struggle with unappropriated tools or methods, often obtaining inaccurate if not erroneous results. They may spend a lot of time in implementing preliminary data analysis pipelines for the sole objective of exploring the

The associate editor coordinating the review of this manuscript and approving it for publication was Michele Magno.

characteristics of datasets, before delving into more sophisticated analyses.

To mitigate these difficulties, in our previous work - focused on computational genomics - we designed and implemented GeCoAgent [5], a conversational agent to empower biologists with limited computational skills. GeCoAgent is a web application that comes with a large integrated warehouse of more than 200,000 genomic experiments. The user can interact with a conversational agent to explore these information and to progressively request the data she is interested to. A user study, involving Ph.D. candidates in biology, biochemistry, and computational biology, showed that even users with limited computational background can succeed in performing data science tasks using GecoAgent, far beyond what they could have achieved by using Python or R. Capitalizing on the experience of this past research, we have developed **DSBot**, an interactive machine learning tool that combines Natural Language Processing, conversational technologies, and AutoML techniques. The objective of DSBot is to translate a research question, expressed in natural language, into an executable data science pipeline on *any* dataset. Our system is based on four design principles that extend and improve GeCoAgent in several directions:

- DSBot is *domain-independent*: it is decoupled from any data repository and operates on any tabular dataset, enabling the user to upload their own data. As such, DSBot can be exploited for any arbitrary Data Science application. In contrast, GeCoAgent is domain-dependent: it operates on its own genomic data warehouse only, and its analysis capability is specialized for this specific content;
- DSBot requires even lower knowledge on data science methods than GeCoAgent, and enables any domain experts to perform complex data analysis tasks on their own data.
- GeCoAgent requires the user to provide a procedural specification of the data analysis pipeline, i.e. the user must progressively specify the operational steps needed to perform the desired analysis. In DSBot, the user expresses their research question in a declarative way, i.e., describing the analysis goals and not the operations or algorithms needed to build the desired results. For example the user can ask *"What factors influence most the price of a house?"* rather than specify how to obtain the results, e.g., *"Encode categorical variables, apply a scaler, train a linear regression model and return the features with the highest absolute value of the associated coefficient"*. DSBot automatically translates a user's declarative requirements into an operational pipeline, choosing the best algorithms and parameters to optimize the results.
- From a conversational design perspective, in GeCoAgent the conversation is driven only by the user's choices selected among a set of pre-defined options provided at each step by the conversational agent. In DSBot, the conversation is generated by taking into account both the user's explicit choices and the dataset properties. The conversational agent interacts with the researcher proactively and in a participatory way, both during the elicitation of data analysis requirements and during the progression of the pipeline execution. For example, some portions of the dialogue are devoted to check with the user if the agent has understood the user's intentions correctly. In addition, DSBot involves the user in the salient decision points where knowledge about the semantics of the uploaded data and human's explicit choices are needed. For example it asks which features to select (''*Please list the features you want to consider*'') or how to deal with missing or noisy data (''*Should out-of-range values be removed because most likely a measurement error, or should they be considered as acceptable?*'').

DSBot is an end-to-end system, able to assist the user in the whole data analysis process from beginning (research question declaration) to end (analysis results reporting). Once users have uploaded their dataset and expressed their data analysis need (''research question''), the system analyzes the data and takes care of pre-processing operations, for example, by transforming categorical variable with one-hot-encoding representation for clustering analysis, or normalizing quantitative variables, or addressing missing value issues. Once all the necessary information has been collected from the user and from the data, DSBot exploits a custom-made Automatic Machine Learning (AutoML) algorithm to select the best algorithm and tune its (hyper)parameters. In the end, DSBot returns graphs and tables that summarize the analysis results and are integrated with comments in natural language.

We tested DSBot on 100 ''research questions'' to assess its capability to translate user's information needs into correct operational pipelines. We also evaluated the execution time and outcomes of the analyses performed by DSBot over 30 datasets of different nature, and compared them with the ones obtained using TPOT, a well-established AutoML tool [6]. Our results show that our system achieves comparable performances (in terms of accuracy and root mean square error) with a significantly shorter execution time.

The innovation presented in this article is concerned with not only the delivery of a new domain independent tool to assist inexperienced users in performing data science analyses, but also a novel approach that combines Large Language Models, Conversational Technology, and AutoML techniques in a sophisticated unique way. Large Language Model techniques are used to translate the user's research questions, expressed in a declarative way, into operational specifications, i.e., the operations and algorithms that compose the data analysis pipeline. Conversation Technology is exploited to engage users into a dialogue devoted to validate with them the correctness of the operational pipeline w.r.t. their needs, using concepts and terminology that can be understood by people with low data science knowledge. Conversational Technology is also exploited during the execution of the pipeline when it is necessary to collect further information from the user. AutoML techniques are used to select the 'optimal' ML algorithm by: i) executing multiple ML algorithms on subsets of the dataset uploaded by the user; ii) automatically selecting the best one along with the values of its (hyper)parameters; iii) running the selected algorithm on the full dataset.

II. STATE OF THE ART

A. AUTOMATIC CODE GENERATION

Coding is a cognitively expensive task [7] in which programmers must first learn the programming language and then translate their ideas in the language learnt [8], [9]. A vast amount of research has tried to develop interfaces that translate natural language directly into executable code.

Today, Automatic Code Generation tools vary a lot in their functioning, accepted input, and programming language produced. Authors in [10] propose a taxonomy for classifying these applications, according to the input type – a high-level description of the task to be executed or a detailed description of all the commands to be programmed – and output to be produced – whether executable code, code snippets, or a representation in an intermediate language.

From the technological perspective, we can cluster Automatic Code Generation tools into three main groups. The first one includes simple instruments driven by grammars, matching natural language patterns and translating them into executable code [11]. The second one includes more complex systems, using probabilistic or combinatorial grammars to enrich the set of user sentences accepted [12], [13], or exploiting natural language processing techniques to understand users' requests and extract useful information for the generation of the code [14], [15]. A third most recent group exploits machine learning techniques to automatically generate executable programs. In particular, Neural Networks are widely used for this purpose, together with large corpus of training data [16], [17], [18].

B. AutoML

Automated Machine Learning (AutoML) is a branch of artificial intelligence that aims at automatizing the entire machine learning process [19]. Two categories of users benefit from AutoML: data scientists, who can concentrate their focus in models optimization and interpretation, and non-machine learning experts, who have an easier access to machine learning methods [19]. Three widely used AutoML systems are Auto-WEKA, Auto-Sklearn, and TPOT.

Auto-WEKA [20] automatically selects the best algorithm and configuration between the ones offered by Weka platform. The choice is done by transforming the problem of choosing algorithm and parameters into a bayesian optimization problem. Auto-WEKA is agnostic from the optimization technique: it can operate either by choosing the algorithm and its hyperparameters consequentially, or simultaneously.

Auto-Sklearn [21] is an AutoML library that operates on scikit-learn. It improves its performances thanks to additional steps in the optimization pipeline, a meta-learning phase at the beginning of the process to warm-start the bayesian optimizer, and an ensemble construction mechanism that combines models evaluated during the optimization.

TPOT [6], exploits genetic programming as optimization engine. Machine Learning pipelines are represented as tree structures on which the genetic algorithm is executes. Every pipeline is evaluated, and the top performing ones are used to create the next generation of pipelines.

While automation and efficiency are among AutoML's primary features, the process still needs human intervention at a number of critical phases, such as identifying the relevant features of domain-specific data or picking the appropriate machine learning problem [22].

C. INTERACTIVE MACHINE LEARNING

With the advances in ML and Data Science, we have witnessed an increasing interest in improving Data Science tools in order to reduce the effort of expert data scientists and to facilitate advanced data analysis for non-experts, promoting accessibility to and adoption of Data Science solutions. In [23] and [24], the authors highlight the need for ML methods and tools that are more interactive and better integrated with the human expertise and needs, complementing and enhancing the work of domain experts, particularly in situations where providing fully automated functionality is computationally very demanding. In the current state of the art, a number of interactive ML platforms exists that we can categorize according to the degree of freedom they leave to users.

The simplest platforms support the execution of a single machine learning task, typically classification. Users must only upload data with some additional information (such as the label variable, in the case of supervised learning) and the software automatically performs the analysis and builds the model. In [25], the author proposes a web interface to create a multi-label image classifier built on TensorFlowJS [26]. Uploading the image files in different folders for every label, the system produces a Convolutional Neural Network and produces two files, one containing the architecture of the network, the other its weights. Teachable Machine is a platform provided by Google to create images and audio classifiers [27]. Users upload samples and through the click of a single button and the platform trains a classification algorithm to solve the given problem. Then, users can export the model as a snippet of JavaScript code to be employed in any project. Iyer et al. propose Trinity, a web interface to analyze spatial data [4], automatically creating binary and multi-classes classificators. Data are pre-processed and prepared for CNN-based learning, and visualizations are returned to users. If the output of model is satisfactory, Trinity offers a workflow to put it into production.

Other tools sacrifice the complete automation of the process and let users choose the best performing algorithm by confronting the solutions proposed by the platform. For example, Model LineUpper combines visualizations and Explainable AI techniques to interactively compare AutoML solutions [28]. Distilling the results of an empirical evaluation of the system, the authors elicit a set of guidelines useful for the design of platform for comparing Data Science models. All the guidelines focus on the importance of the freedom offered to the user in adjusting models and the transparency of the operations, such that users can understand precisely what the system did automatically.

Other systems help users to identify the appropriate operations for the analysis they want to perform. For example, Snowcat [29] automatically proposes a set of research problems to answer through the data to be analyzed. Based on the user's problem choice, it trains a set of models and provides an interactive dashboard to explore them. Users also have the possibility of downloading the generated models for further analysis.

In AutoDS [30], once data workers have uploaded their dataset, the system automatically suggests ML configurations, preprocesses data, selects algorithms, performs model training, and then presents the resulting pipeline on web-based graphical user interface and a notebook-based Python programming interface. The paper reports an empirical controlled study which explored AutoDS with 30 professional data scientists; one group used AutoDS, and the other did not, to complete an assigned data science project. The results showed that AutoDS improved productivity, and the models produced by the AutoDS group had higher quality and less errors. Still, the human confidence on the final model was lower in the AutoDS group. Lack of total control on the system is considered to be the predominant cause of this skepticism. In addition, 43% of participants declared that they trusted AutoDS (i.e., they were confident in the system and considered it reliable (13% did not, and 43% were neutral). Lastly, 50% of participants did not believe that AutoDS would replace human data scientists (only 10% had this belief, with the rest remaining neutral.)

In [31], the authors developed a visual method to compare multiple classifiers considering model performance, feature space, and model explanation. ModelWise adapts visualizations with rich interactions to support multiple workflows to achieve a model diagnosis, improvement, and selection.

Many tools concentrate on offering users a set of instruments they can use for their analysis, at the cost of requiring users to have a good understanding about the methodologies they want to use. For example, TwoRavens is an interface to operate on data publicly available on Dataverse repositories [3], [32]. Through a graph-based UI, users can explore the data they selected and choose the statistical method to analyze them.

Pyrus, is an online modelling environment developed for authoring data science pipelines through a graphical interface [33]. It is designed around the principle of separation of concerns: data scientists can implement block units that perform data science operations in a dedicated interface, while domain experts can use a block interface to create pipelines with the units implemented in the system. Still, to use this platform users must having a basic understanding of data science to better compose their pipelines.

Some studies explore the use of conversational technologies during the data science process. Ava [34] works on a structured process: the conversation predicates on a pre-defined process in which the conversation asks users the desired operations and parameters. Yet effective, this choice constraint users to use only the modules that fit in the process model. Iris [35], instead, acts as a conversational wrapper for data science operators that allow users to compose their pipelines in freedom. Yet, users must know the modules and their functionalities; the conversational layer does not offer support in composing the operations.

In summary, interactive machine learning is an emerging and prolific field of research. At the same time, our literature review shows that users must have good expertise in order to fully trust the results produced by these platforms [30], [36]. With our work, we aim at filling this gap, providing a tool that does not require advanced Data Science knowledge, leaves users freedom to perform analysis driven by research questions, and provides enough information and explanation for enhancing the user's trust in the results.

III. METHODS

Data Science is an extremely broad topic, encompassing a wide set of research questions and possible analyses, some of which requires *ad-hoc* solutions; DSBot operates on a well defined subset of tasks (and therefore analyses) and under a number of constraints. The input to DSBot is limited to a single table, where each sample, i.e., row of the table, is a tuple of features, each one independent from the others. Any spreadsheet or result of DBMS query can be used. Sorting and relationships between rows of the dataset (such as time series) are not represented, and therefore the methods that exploit these properties cannot be applied. The dataset can have any number of features, but at most one target. Moreover, the type of data of both feature and target columns are limited to real numbers and categories, thus excluding multidimensional or complex information such as images, audio tracks or text; such complex data require domain specific pre-processing and feature engineering, that are beyond the scope of DSBot.

A. OVERVIEW OF THE SYSTEM

We present an overview of the DSBot mechanism used to build and execute data analysis pipelines. In order to complete an analysis task, DSBot goes through several stages; some of them require interactions with the user, while other phases are fully automatized. The whole process is illustrated in Figure 1 and comprises the following eight main steps:

1) The user uploads the dataset and specifies the target column;

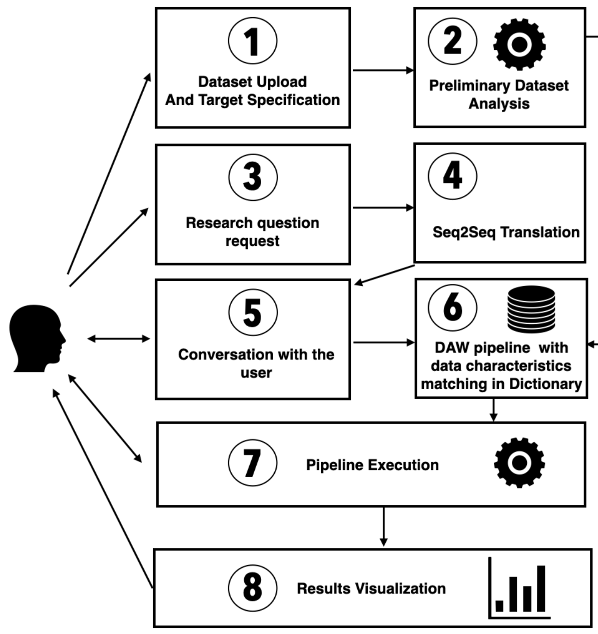

1
Dataset Upload And Target Specification
2
Preliminary Dataset Analysis
3
Research question request
4
Seq2Seq Translation
5
Conversation with the user
6
DAW pipeline with data characteristics matching in Dictionary
7
Pipeline Execution
8
Results Visualization

FIGURE 1. Conceptual architecture of the system.

2) The system runs a set of standard analyses on the dataset to infer descriptive characteristics, such as data types or the presence of missing values;
3) The user formulates a research question as a natural language sentence;
4) The system applies a machine translator to translate the natural language question in a Data Analysis Workflow (DAW) pipeline;
5) A conversational agent engages the user in a dialogue to ensure that the produced DAW sentence corresponds to the user expectations; in this step, DSBot may also use the dialogue to elicit other requirements from the user, in order to correct or refine the DAW pipeline;
6) The confirmed DAW pipeline is compared with a pipeline dictionary, from which the best matching pipeline is selected; the pipeline produced in this way can be augmented with additional operations, to cope with the dataset characteristics (e.g., handling of missing data and/or outliers);
7) The pipeline is executed; during execution, the system may interact with the user to drive the execution flow, for example asking for a specific subset of the features upon which the analysis should be performed;
8) The results of the analysis are visualized.

To recap, starting from a declarative specification of the analysis by the user, which means a high-level description of the desired output that abstracts from any operational details, DSBot analyzes the input dataset to produce and execute an appropriate pipeline that matches the user goals.

B. COMPONENTS

Hereafter, we present and discuss the details of the various components of DSBot and show how they interact.

TABLE 1. High level and low level symbols included in DAW domain specific language.

High Level	Low Level
`missingValues`	`fillMissingValues`
	`removeMissingValues`
	`missingValuesHandle`
`encoding`	`oneHotEncoder`
`outliers`	`outliersRemove`
	`outliersDetection`
`zeroVariance`	`zeroVarRemove`
`strongCorrelatedFeatures`	`correlatedFeaturesRemove`
`featuresToRemove`	`removeFeatures`
`preprocessing`	`standardization`
	`normalization`
`labelOperations`	`labelRemove`
	`labelAppend`
`correlation`	`pearson`
	`spearman`
`classification`	`autoClassification`
	`randomForest`
	`logisticRegression`
	`kNeighbors`
	`adaBoost`
`clustering`	`kmeans`
	`dbscan`
	`agglomerativeClustering`
`outliersDetection`	`outliersDetection`
`featureSelection`	`lasso`
	`selectKBest`
	`laplace`
	`userFeatureSelection`
`featureImportance`	`featureImportance`
`featureEngineering`	`pca2`
	`mds`
`associationRules`	`apriori`
`regression`	`autoRegression`
	`ridgeRegression`
	`linearRegression`
`performance`	`regressionPerformance`
	`confusionMatrix`
`plot`	`scatterplot`
	`clustermap`
	`roc`
	`lassoPlot`
	`tableRegression`
	`tableAssociationRules`
	`featureImportancePlot`

1) DATA ANALYSIS WORKFLOW DOMAIN-SPECIFIC LANGUAGE

The Data Analysis Workflow (DAW) is a Domain Specific Language (DSL) that encodes the pipelines for Data Analysis. It is a formal language that aims at representing the sequence of data manipulation and analysis operations for (a) being interpreted and executed and (b) being stored and searched in the knowledge base. A DAW sentence is composed of two parts: the dataset descriptions and the list of operations. The first is a set of terms describing the main dataset features, such as `missingValues`, `outliers`, `zeroVarianceFeatures`, the second one denotes a sequence of operations to be performed on the dataset to accomplish a specific objective. DAW language is extensible; new terms can be added to both datasets description and workflow operations as new features are added to the system. A workflow description in DAW is a sequence of symbols that represent the linear flow of operations to be

TABLE 2. Dataset Characteristics inferred by DSBot.

Name	Description
`missingValues`	Some values are missing or NA
`categorical`	Has both categorical and numeric features
`onlyCategorical`	Has only categorical features
`continuosLabel`	The target has continuous values
`categoricalLabel`	The target has categorical values
`outliers`	Some features presents outliers
`lessThan3Features`	Less than 3 features are present
`strongCorrFeatures`	Presence of strongly correlated features
`uninformativeFeatures`	Some feature is not informative
`zeroVarianceFeatures`	Presence of features with zero variance

executed. There are two classes of symbols, namely *high-level* and *low-level*, which are organized in a hierarchy in which each low-level symbol is a specialization of a high-level one. Low-level symbols have a one-to-one correspondence with a operation that can be directly executed, while high-level symbols needs to be first specialized in a low-level symbol, either by automatic mechanisms (described later in this Section) or by interacting with the user. A comprehensive list of the symbols of DAW is reported in Table 1; a brief description of operations associated with low-level symbols is in the Supplemental Material. As an example, consider the following DAW sentence:

```
userFeatureSelection oneHotEncode classification roc
```

This sentence describes a data analysis workflow applicable to a dataset. The four operations are executed sequentially; `userFeatureSelection` is a low-level operator that can be executed and it may require to interact with the user to ask for further information. On the contrary, `classification` is high level and, before being executed, it must be specialized to a low-level operator, by means of a mechanism described later. The remaining operations are low-level and do not require user interaction, thus they can be executed automatically.

DAW has two main uses: describing the analysis to be performed and storing manually curated models of analyses in the pipeline dictionary. Note that the symbols of DAW correspond to a specific algorithm and abstract from its parameters; such parameters are automatically tuned by the pipeline executor. Finally, it is worth mentioning that the language is extensible; new symbols can be added to DAW as new features are added to the system.

The DAW also provides a further benefit: it conceptually and logically separates between the production of the analysis and its execution. As a consequence, if better tools are available to perform Data Science operations, it would be sufficient to substitute the execution engine, without affecting the translation machinery. For example, one could provide a big-data version of DSBot simply replacing the current execution engine with one based on the ML libraries of Apache Spark.

2) PRELIMINARY DATASET ANALYSIS

Once the user has uploaded the data and has specified the label, DSBot proceeds automatically to infer the characteristics of the current dataset to drive the selection of possible analyses and the choice of a good pipeline. The list of characteristics is reported in Table 2. While many of them are self-explanatory other require further description:

- `outliers`: indicates that the dataset contains items whose values significantly differ from the others. Formally, the input dataset $D \in n \times p$ contains at least one row $d_i, i = 1, \ldots, n$ with more than the 90% of the numerical attributes of d_i such that:

$$|d_{i,j} - \mu_j| > 3 \times \sigma_j$$

  where $d_{i,j}$ is the element of the dataset at the i-th row and j-th column, μ_j and σ_j are respectively the mean and the standard deviation of the j-th column;
- `strongCorrFeatures`: indicates the presence of pairs of numerical columns (d_i, d_j), $i \neq j$ with a Pearson correlation coefficient greater than 0.9;
- `uninformativeFeatures`: indicates that the dataset contains at least one categorical column such that the number of its distinct elements is larger than half of the number of rows of the dataset. In other words, each value, on average, is associated to less than two samples and thus it is likely to be an identifier of the samples (e.g., patient id).

The above mentioned characteristics allow the system to select different pipelines that are most appropriate for the uploaded dataset. Indeed, each characteristic is handled by different operations included in the available pipelines within the dictionary, explained in Section III-B5.

3) QUESTION TRANSLATION

In order to translate from English to DSL (domain specific language), we needed a model that could perform well even with a low resource language. After evaluating various options, we decided to use GPT-2 (Generative Pre-trained Transformer 2) as our machine translation model. GPT-2 is a pre-trained transformer-based language model developed by OpenAI that has been trained on a massive corpus of text data. This makes it highly adaptable and able to be fine-tuned for a variety of natural language processing tasks, including machine translation.

One of the key features of GPT-2 is its use of a self-attention mechanism. This allows the model to weigh the importance of different parts of the input text and understand the sentence's context, generating more accurate translations.

To fine-tune GPT-2, in particular GPT-2 355, we used a synthetic dataset of around 353,000 sentences that were generated from a set of manually created templates with their corresponding translation in DAW.

Since the tokens in DAW language are a defined set, sentences produced by GPT-2 were filtered to ensure that they contained only words that are contained in DAW vocabulary.

4) CONVERSATIONAL COMPREHENSION ASSESSMENT

After the conversion of the user's request in a DAW, a conversational agent assesses whether the system has correctly interpreted the input sentence. To do that, the system receives the DAW sentence and converts the operations contained in it into a textual description. Descriptions are merged in a single textual message that is sent to the user asking for confirmation; the text describes operations at a high level, abstracting from their technicalities and focusing instead on their expected result. Data Science jargon is not used, as it may be not understood by DSBot users.

Being interested in the outcome of the operation and not in the functioning of the algorithm itself, we can use the same textual description for different terms that belong to the same algorithmic family. For example, we can transform both "`kmeans`" and "`agglomerativeClustering`" modules in the following description: "*to group your data in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).*". The same description holds for "`clustering`" term.

Hence, the symbols in the DAW belong to high and low level classes. Each high level class symbol corresponds to one or more symbols in a low level class. For example, the symbol in high level class *classification*, corresponds to low level symbols such as *randomForest, logisticRegression, autoClassification*, etc.

Every node in the tree may have a textual description, that contains the sentence to be produced in the conversation. When a term must be translated, the system retrieves, through a tree-search, the deepest node having a textual description in the path from the root to the searched node, and returns that description, which is then concatenated to the descriptions of the other words in the DAW, and sent to the user for confirmation.

Users can confirm the textual description, or ask for more detailed explanations, or for an example of application of the workflow, so as to understand if they have correctly understood. Explanations and example production follows the same principles of the textual one. If users confirm the workflow, the control is passed to the Workflow Enrichment module (Sec. III-B5).

If the system has not correctly understood what the user wants to do, the conversational agent guides the user in the selection of an operation, following the state-machine-based representation of the conversation flow shown in Figure 2. Rounded corners rectangles represent the moments in which the conversational agents sends a message to the user through the chat, and waits for one of the responses indicated on the exiting arrows; diamond shapes represents the agent's decisions on dataset properties; rectangles represents decisions on the data science pipeline that will be proposed to the user.

The conversation aims at eliciting the user's operational goal, i.e., the high-level operation the user wants to perform: clustering, regression, classification, association rules, or correlation matrix. The conversation exploits the dataset information to improve the experience and facilitate users' comprehension. If the dataset has a label, the first proposed operation is the prediction of a value; in case of affirmative response from the user, the system automatically decides whether applying regression or classification according to the nature of the label. If, instead, the user has not indicated any label, then the conversational agents first asks whether the users wants to find relationships in the data (i.e., association rules or correlations), clustering, or prediction tasks.

When the family of algorithms is identified, heuristics on data are used to elicit the algorithm to use. For example, once the user agrees on finding relationships in the data, correlation is automatically chosen if the dataset only contains numerical variable, while it is excluded if the dataset does not continue any numerical variable. In the same way, in prediction tasks, classification or regression are chosen according to the nature of the variable to predict. When the desired operation has been elicited, a new pipeline containing the operation is produced and the control is passed to the Workflow Enrichment module.

5) PIPELINE DICTIONARY AND WORKFLOW ENRICHMENT

As illustrated in Figure 1, the DAW obtained as translation of the research question, together with the dataset characteristics, is matched against a dictionary which includes manually curated pipelines. The resulting best match is then used to correct and augment the DAW, taking advantage of established best practices in Data Science. For example, if the dataset contains columns that have zero variance (i.e., whose values are constant), the best match includes zeroVarianceRemoval step. Up to now, the pipeline dictionary contains 9634 pipelines distributed over 439 combinations of pipeline characteristics. By design, the pipeline dictionary is extensible with new pipelines.

Consider a input dataset D, with the set of characteristics $\{ds_1, \ldots, ds_n\}$ and a user's question translated into a sequence of operations $op_{u_1}, op_{u_2}, \ldots, op_{u_U}$. The search in the pipeline dictionary is meant to identify the entry:

$$(ds_1, \ldots, ds_m)op_{k_1}, op_{k_2}, \ldots, op_{k_U}$$

with the constraint that:

$$\{ds_1, \ldots, ds_n\} \subseteq \{ds_1, \ldots, ds_m\}$$

(i.e., every characteristic of the dataset in the pipeline dictionary entry must be found in the input dataset). Each compatible pipeline in the pipeline dictionary is then ranked with a matching score and the most fitting one is then chosen.

In order to identify such best matching sequence, we implemented dynamic programming algorithm inspired by the Needleman-Wunsch algorithm for the pairwise optimal alignment of sequences [37]. This algorithm will select the best matching pipeline based not only on the user's input, but also on the characteristics of the dataset, therefore improving the outcome of the analysis. An example of analysis results before and after the enrichment of the pipeline is reported in Figure 3, considering a scenario in which the user uploaded the Penguin dataset [38] and asked for a clustering analysis. Figure 3(a) shows the

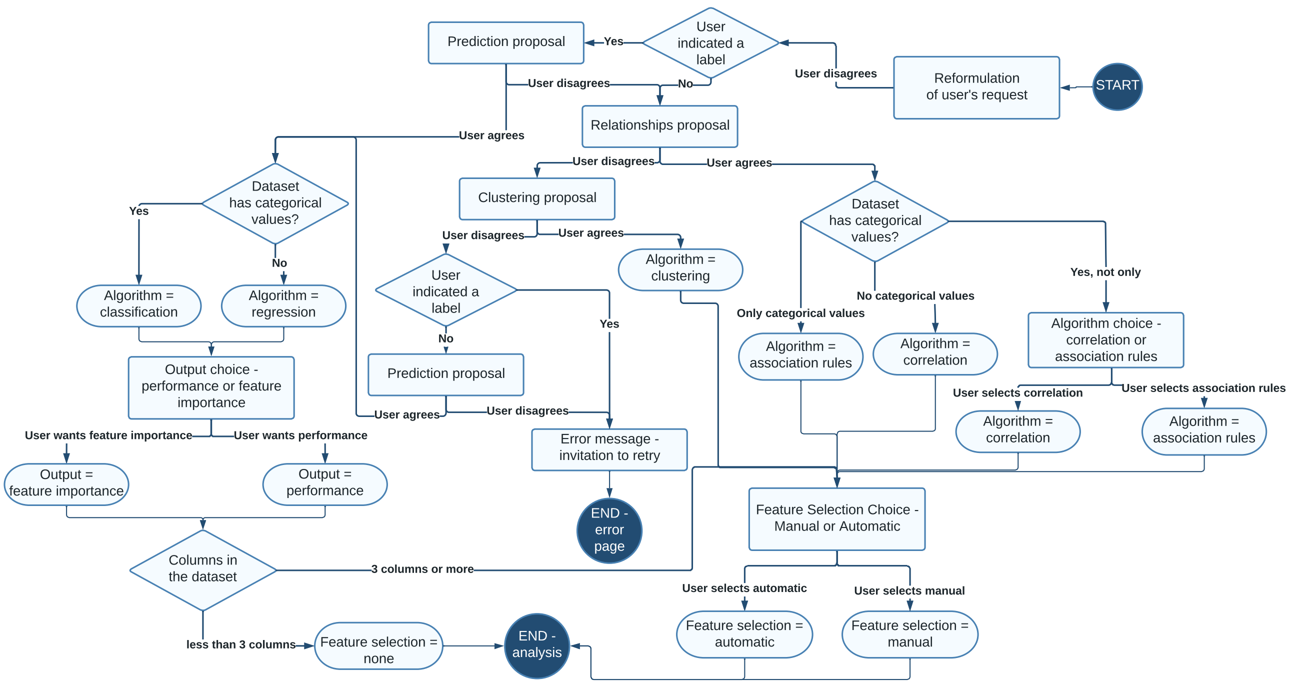


FIGURE 2. Finite State Machine of the high level conversation flow for user's operational goal elicitation.

results of the pipeline execution when the dataset characteristics are *not* considered. In this case, the pipeline executed is `removeMissingValues`, `oneHotEncoder` and `kmeans`. Figure 3(b) shows the results of the second analysis, which takes into account the characteristics of the dataset and is able to extract more significant clusters. In this case, the pipeline executed is `fillMissingValues`, `oneHotEncoder`, `normalization` and `kmeans`.

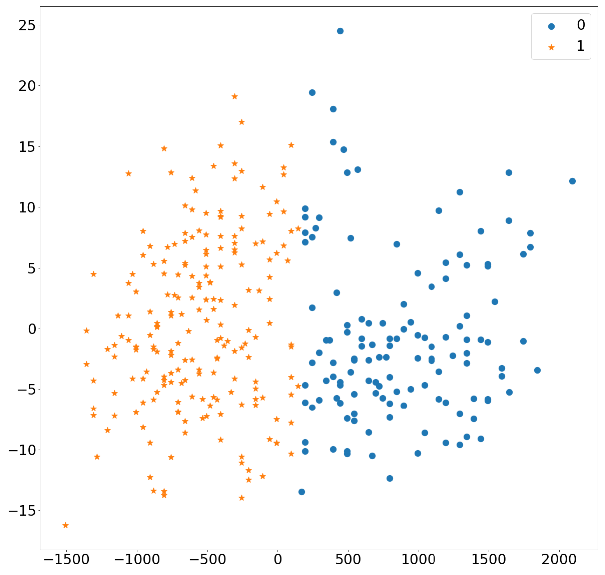
6) AUTOCLASSIFICATION AND AUTOREGRESSION MODULES

In DSBot, not all the modules require input from the user. There are some operations that are fully automated: the most relevant ones are IRAutoClassification and IRAutoRegression. These two execute different modules of respectively classification and regression, tuning the parameters. The module and the parameters with the best accuracy and root mean squared error (RMSE), respectively, are chosen for the analysis and are used for the prediction.

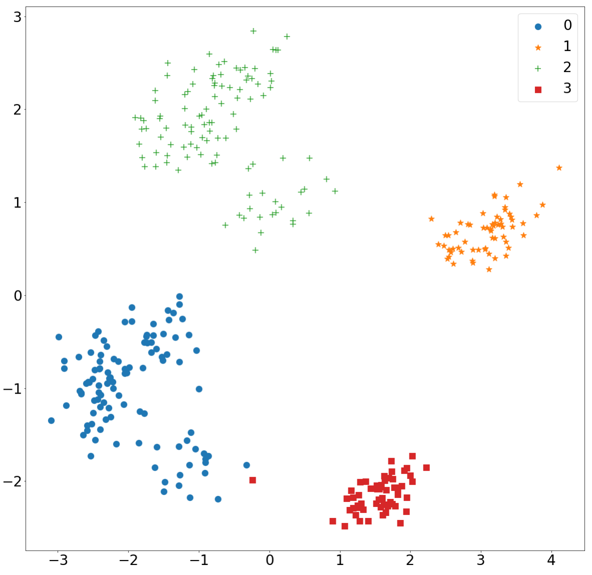
More precisely, the AutoClassification module is applied after some preprocessing operations and a Lasso feature selection. It divides the dataset into training and test set and it runs four different modules of classification: a Random Forest classifier, an Ada Boost classifier, a k-nearest neighbors (KNN) classifier and a Logistic Regression classifier. In order to decide which module is the best one, it also runs a parameter tuning module on each of them to try the best combination of parameters and compare the four modules each with the best parameters. In particular, it applies a random search strategy for the four modules, each one on its own parameters. The search starts by evaluating all the candidates (i.e., combination of parameters) with a small amount of samples and selects the best combination of parameters iteratively, using more and more samples. The candidates for each modules are listed below.

- Random Forest Classifier:
  - criterion: Gini or entropy;
  - number of estimator from 10 to $min(max(\sqrt{n\_row * n\_col}, 50), 500)$, with a step of 10;
  - min_samples_split: from 2 to $min(\frac{n\_row*3}{2}, 100)$ with a step of 5;
  - max_depth: $[2, \sqrt{n\_col}, \frac{n\_col*3}{2}$, None].
- Logistic Regression:
  - inverse of regularization strength: [1e-4, 1e-3, 1e-2, 1e-1, 0.5, 1., 5., 10., 25.].
- K-nearest neighbors:
  - n_neighbors: from 1 to $min(\frac{n\_row}{10} - 1, 50)$ with a step of 1;
  - weights: uniform or distance;
  - p: 1,2.
- Ada Boost:
  - number of estimator from 10 to $min(max(\sqrt{n\_row * n\_col}, 50), 500)$, with a step of 10;
  - base estimator: [DecisionTreeClassifier, ExtraTreeClassifier, SVC].

After having computed the accuracy for each module and for different sets of parameters, the AutoClassification module



(a) Results of the execution of the pipeline chosen without considering the dataset characteristics.



(b) Results of the execution of the pipeline that takes into account the dataset characteristics with the selected workflow.

FIGURE 3. Example of an analysis' results before (a) and after (b) the enrichment of the pipeline.

defines the best combination according to the accuracy obtained and run it on the training set. It then saves the prediction of the testing sets and the features importance for showing the user either the performance, with or a ROC curve or a confusion matrix, or the importance of the features in two different plots.

Also, an AutoRegression module is applied after some pre-processing operations and a Lasso feature selection. It divides the dataset into training and test set and it runs four different modules of regression: a Random Forest Regressor, an Ada Boost Regressor, a Linear Regressor and a Ridge Regressor. In order to decide which module is the best one for that dataset, it not only runs these four modules, but also runs a parameter tuning module on each of them to try the best combination of parameters and compare the four modules with the best parameters. Also in the regression case, we tuned the parameter using a random search with cross validation that starts considering all the possible combinations of parameters on a subset of samples, and then iteratively selects the best combination applying it to a bigger subset of samples. The parameters for the regression modules are:

- Random Forest Regressor:
  - number of estimator from 10 to $min(max(\sqrt{n\_row * n\_col}, 50), 500)$, with a step of 10;
  - min_samples_split: from 2 to $min(\frac{n\_row*3}{2}, 100)$ with a step of 5;
  - max_depth: [2, $\sqrt{n\_col}$, $\frac{n\_col*3}{2}$, None].
- Linear Regressor.
- Ridge Regressor:
  - alpha: from 0 to 1 with a step of 0.1.
- AdaBoost Regressor:
  - number of estimator from 10 to $min(max(\sqrt{n\_row * n\_col}, 50), 500)$, with a step of 10;
  - base estimator: [DecisionTreeRegressor(), ExtraTreeRegressor(), SVR(kernel='linear')].

After having computed the accuracy for each module and for different sets of parameters, the `AutoRegression` module defines the best combination according to the root mean squared error obtained and run it on the training set. It then saves the prediction of the testing sets and the features importance for showing either the performance or the importance of the features in two different plots with a little explanation of the results.

C. ARCHITECTURE

The main components of the system are shown in the architecture in Figure 4.

- The *frontend* allows the user to interact with the tool in a friendly way. It consists of a single-page web application, with different modules for the web chat, the input acquisition and the result visualization; it has been implemented using Vue.js framework to guarantee modularity and extendability.
- The *backend* comprises several components, the most relevant of which are the *query translator*, the *pipeline dictionary*, the *pipeline executor* and the *dialogue manager*.
  - The backend receives the dataset and the *pipeline executor* executes the preliminary analysis on the data.
  - The *query translator* transforms the research question into a DAW pipeline.

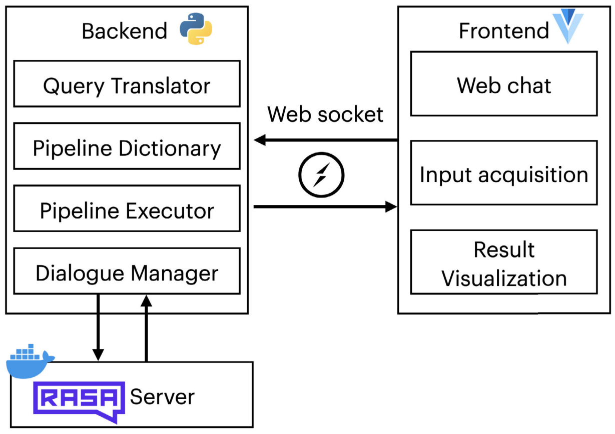


FIGURE 4. Architecture of DSBot.

- The *dialogue manager* checks with a short conversation with the user if the translation is correct and helps the user if she does not understand what she has to do.
- The *pipeline executor* looks for the best pipeline in the *pipeline dictionary*, creates a module for each operation it has to execute and run it. To complete the pipeline, the *pipeline executor* can require human intervention and ask the user for some parameters or details. Other operations instead perform automatically the analysis, two examples are IRAutoClassification and IRAutoRegression, detailed in Section III-B6. While executing some operations, the *pipeline executor* can sometimes notify the user by providing her with interesting details highlighted during the execution. An example is the percentage of removed outliers.

• The *docker server* is used with RASA, an open-source Natural Language Understanding Unit (NLU) [39]. This service is in charge of translating user sentences during conversation in symbols understandable from the dialogue manager (intents) and extract the parameter necessary for the task completion (entities).

The backend is implemented in Python using the Flask framework to serve the frontend and manage users' sessions. The analysis, and visualization function are fully implemented in Python, leveraging the large availability of libraries for data analysis and visualization. The communication between the frontend and the backend occurs by means of Web Socket (using the socket.io package). This enables fast, real-time, and bidirectional communication; the communication is often instantiated by the backend, which pushes pieces of information to the frontend.

IV. USE CASES

In this section we present two examples of DSBot executions, one on a dataset of genomic features and one on a dataset of clinical and demographic features. The first one shows a full example of analysis starting from the upload of the dataset until the visualization of the results; the second one focuses on the conversational part, and shows a conversation in which DSBot initially fails in understanding the natural language request from the user.

A. ANALYSIS USE CASE

This use case concerns a data-driven analysis of genomic data of patients affected by breast cancer, one of the most common tumor types. Breast cancer is commonly classified in four molecular subtypes, namely *basal*, *luminal A*, *luminal B* and *her2* [40]. Different subtypes influence the development of the disease as well as the choice of the best therapy [40].

We assume as user a clinician who analyzes a genomic dataset containing gene expressions (i.e., the level at which each gene is active within a biosample) for a cohort of 1,127 patients affected by breast cancer in order to understand if there are breast cancer subtypes that are easily confused. Thus, in our dataset, rows refer to the patients and the columns to the genes. For each patient, we measure the expression of the 50 genes of PAM50 panel, which have been identified by oncologists to be the most related with the breast cancer subtype. In addition, each patient is labeled with her *subtype*.

Figure 5 shows the web interfaces for the user to upload the dataset ('pam50_m…fed.csv'), specifying the label 'Expert subtype'. In this phase, the user specifies three characteristics of the dataset: if it has an index column, if it has column names, and the label. DSBot presents a preview of the uploaded table, then analyzes the dataset and extracts the characteristics it needs. Particularly, this genomic dataset has a categorical label and has outliers. These characteristics are used to match the best pipeline according to the user's question.

In the following step, the user expresses a research question using natural language using the interface shown in Figure 6. In the example, the user wants to discover the breast cancer subtypes that are most difficult to discern. His/her question in natural language - reported in Figure 6, could be the following:

Can you tell me which are the most similar subtypes?

DSBot interprets the question and identifies the following preliminary DAW pipeline as appropriate for the user's request:

```
classification confusionMatrix
```

The chatbot provides a short explanation to help the user understand how his/her request has been interpreted, rephrasing the user's request consistently with the preliminary pipeline identified, and then asks for confirmation to proceed (Figure 7- right side). The confirmed preliminary DAW pipeline, along with the inferred dataset characteristics, are used as input for matching the pipeline dictionary. The final DAW pipeline (reported below) is identified and executed, and the final results are visualized to the user ((Figure 7- left side).

```
  labelRemove standardization
outliersRemove
  lasso autoClassification confusionMatrix
```

During the execution of the final pipeline, DSbot provides feedback to the user (e.g. "The 2.838% of the rows are outliers. I will remove them" - (Figure 7- right side). After the final results are visualized, the chatbot highlights the key findings.

B. CONVERSATION USE CASE

In this second use case the user wants to analyze the stroke prediction dataset [41], which comprises clinical and demographic features. Table 3 reports an example of the conversation between the DSBot (B) and the user (U), as it would happen in the case the system fails to interpret the first request from the user. The conversational agent of DSBot provides suggestions on the possible alternative analysis taking into account the characteristics of the dataset. In particular, DSBot proposes to do a prediction analysis since the user indicates a label; in addition, since DSBot sees that the label is categorical, it proposes a classification algorithm rather than a regression one. Subsequently, it wants to know if the required analysis should provide the performances or the importance of the features. In this example, the user requires a feature importance analysis by asking for 'influencing factors'.

DSBot suggests also to do a feature selection analysis before the classification algorithm; in particular, it asks for an automatic feature selection or for a manually one. The user decides for a manually feature selection and provides the list of features to remove. Subsequently, while DSBot performs the analysis, it requires some information from the user, such as removing or filling the missing values. Furthermore, the conversational agent also provides some insights of the analysis, such as the percentage of the removed outliers.

The reported conversation also demonstrates how the tool is domain agnostic; indeed it only considers feature characteristics and not feature semantics to interact with the user and select the best pipeline.

TABLE 3. Conversation between DSBot and the user after the interpretation of the first user's request failed (use case B).

BOT: What do you want to obtain?

USER: analyze my data

B: Let me understand if I interpreted well your request. First, you want to group your data in such a way that objects in the same group (called a cluster) are more similar in some sense to each other than to those in other groups (clusters). Then, you want to visualize the results in a plot. Is it right?

U: I don't know

B: From your request, i understood you want to apply clustering, an analysis that aims at finding groups of data similar each other (clusters). This kind of analysis doesn't require any additional information from you, it works in total autonomy.
Have I understood correctly your request?

U: no

B: I think I misinterpreted your original request. I will ask you some questions to better understand what you want to do. I see that you indicated the presence of a label in your dataset. Do you want to try to predict its value from the other data?

U: yes

B: Given the composition of your dataset, we will use a Classification Algorithm to predict the value contained in column class. Are you interested in the prediction itself, or are you more interested in understanding which are the most influencing factors in determining the prediction?

U: I want to see the influencing factors

B: Ok, we will perform a Feature Importance analysis, to highlight which are the most important factors in the prediction outcome
I need one last piece of information: do you want to use all the columns in your dataset or do you want to use only a subset of them?

U: only a subset

B: Perfect! Do you want me to select the columns automatically?

U: no
B: Ok, I'll ask you the columns later on. Let's proceed! List the features you want to remove using a comma to separate them:
gender, age, hypertension,heart_disease, ever_married,
work_type, Residence_type,avg_glucose_level, bmi,
smoking_status

U: gender, ever_married, work_type, Residence_type

B: Ok, I will consider only columns age, hypertension,
heart_disease, avg_glucose_level, bmi, smoking_status, class
B: The 33.366% of the rows have at least a missing value.
Do you want to REMOVE or to FILL the rows with missing values?

U: Remove

B: Ok, I will remove them. The 20.548% of the rows are outliers. I will remove them

V. EVALUATION

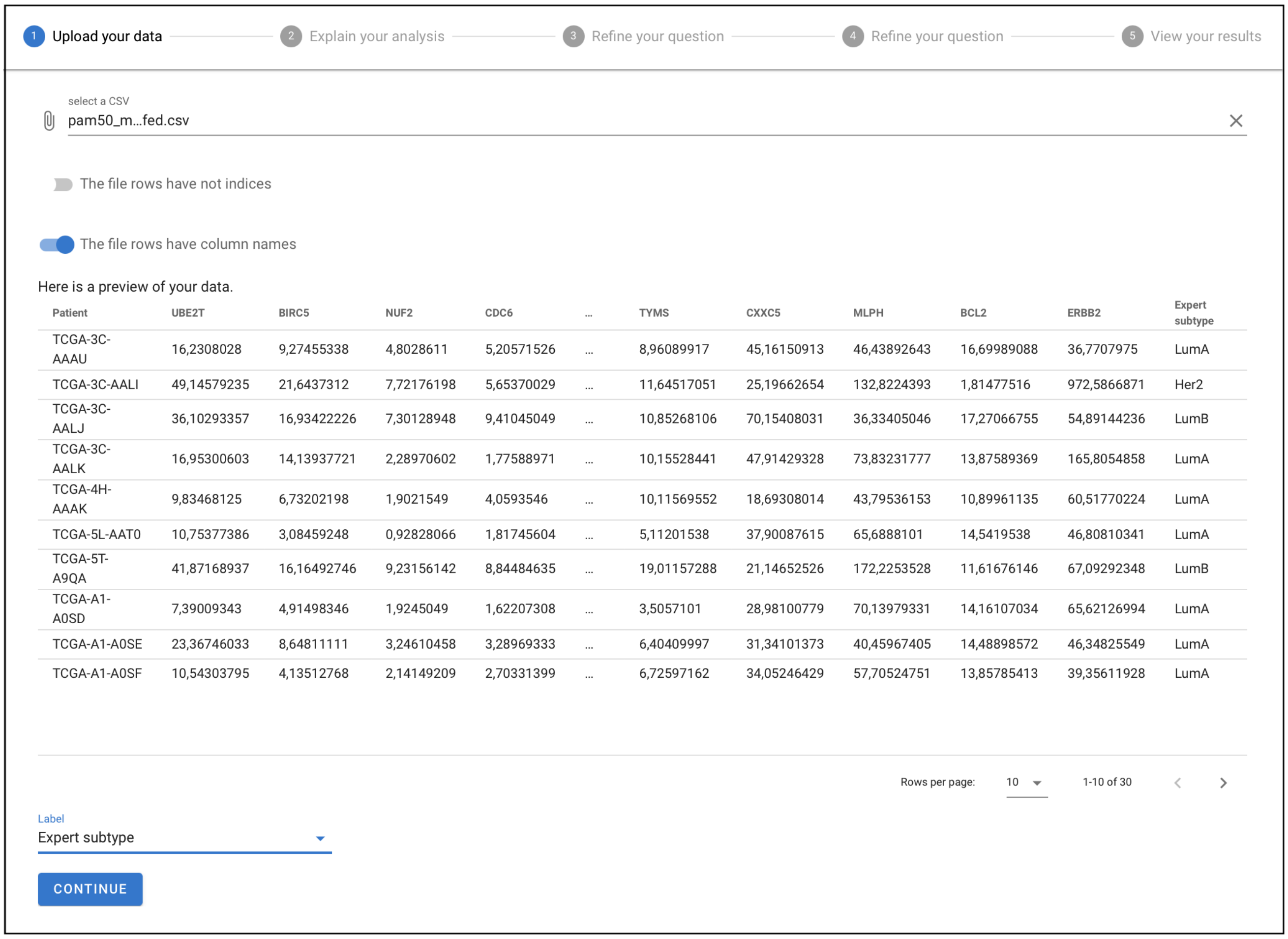
We performed two evaluations assessing different goals: (i) to evaluate the performance of DSBot in terms of accuracy and computation time, and (ii) to test the system capability to understand the user's research questions and to translate them into accurate executable DS pipelines.

A. EVALUATION OF THE AUTOMATIC MACHINE LEARNING PIPELINE EXECUTOR

With this analysis, we wanted to validate the performance of DSBot in terms of accuracy (for classification tasks), and RMSE (for regression tasks). Furthermore, we measured the execution time, which must be minimized to guarantee a smooth user experience.

As a baseline for our experiments, we used TPOT [6], a popular AutoML framework for classification and regression; it uses genetic programming to explore thousands of ML pipelines intelligently and returns the one that optimizes a user-defined score function.

TPOT was chosen as a candidate against which to test the performance of DSBot since also TPOT is implemented in



select a CSV
pam50_m...fed.csv

The file rows have not indices

The file rows have column names

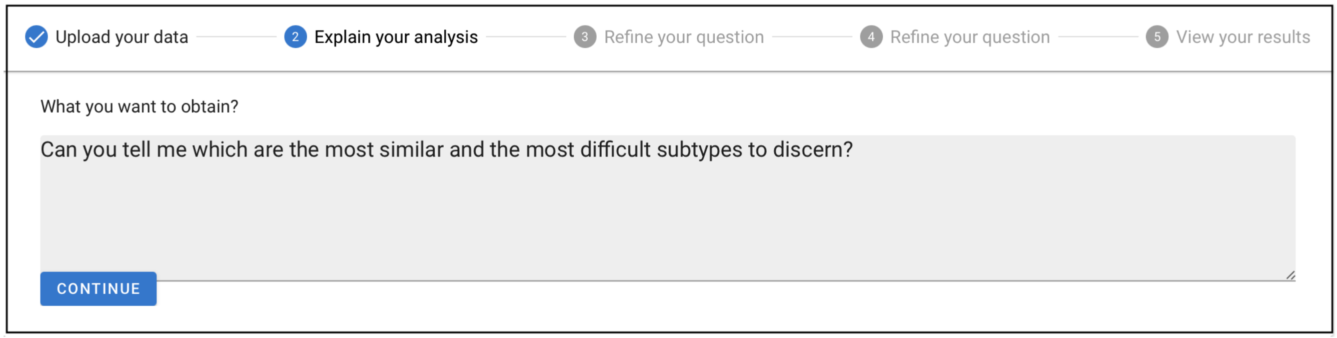
Here is a preview of your data.

Patient	UBE2T	BIRC5	NUF2	CDC6	...	TYMS	CXXC5	MLPH	BCL2	ERBB2	Expert subtype
TCGA-3C-AAAU	16,2308028	9,27455338	4,8028611	5,20571526	...	8,96089917	45,16150913	46,43892643	16,69989088	36,7707975	LumA
TCGA-3C-AALI	49,14579235	21,6437312	7,72176198	5,65370029	...	11,64517051	25,19662654	132,8224393	1,81477516	972,5866871	Her2
TCGA-3C-AALJ	36,10293357	16,93422226	7,30128948	9,41045049	...	10,85268106	70,15408031	36,33405046	17,27066755	54,89144236	LumB
TCGA-3C-AALK	16,95300603	14,13937721	2,28970602	1,77588971	...	10,15528441	47,91429328	73,83231777	13,87589369	165,8054858	LumA
TCGA-4H-AAAK	9,83468125	6,73202198	1,9021549	4,0593546	...	10,11569552	18,69308014	43,79536153	10,89961135	60,51770224	LumA
TCGA-5L-AAT0	10,75377386	3,08459248	0,92828066	1,81745604	...	5,11201538	37,90087615	65,6888101	14,5419538	46,80810341	LumA
TCGA-5T-A9QA	41,87168937	16,16492746	9,23156142	8,84484635	...	19,01157288	21,14652526	172,2253528	11,61676146	67,09292348	LumB
TCGA-A1-A0SD	7,39009343	4,91498346	1,9245049	1,62207308	...	3,5057101	28,98100779	70,13979331	14,16107034	65,62126994	LumA
TCGA-A1-A0SE	23,36746033	8,64811111	3,24610458	3,28969333	...	6,40409997	31,34101373	40,45967405	14,48898572	46,34825549	LumA
TCGA-A1-A0SF	10,54303795	4,13512768	2,14149209	2,70331399	...	6,72597162	34,05246429	57,70524751	13,85785413	39,35611928	LumA

Rows per page: 10 1-10 of 30

Label
Expert subtype

FIGURE 5. **Web user interface to upload the input dataset and indicate the label (use case A).**



What you want to obtain?

Can you tell me which are the most similar and the most difficult subtypes to discern?

FIGURE 6. **Web user interface: textbox for questions (use case A).**

Python and, similar to DSBot, it utilizes the *sci-kit learn* ML library.

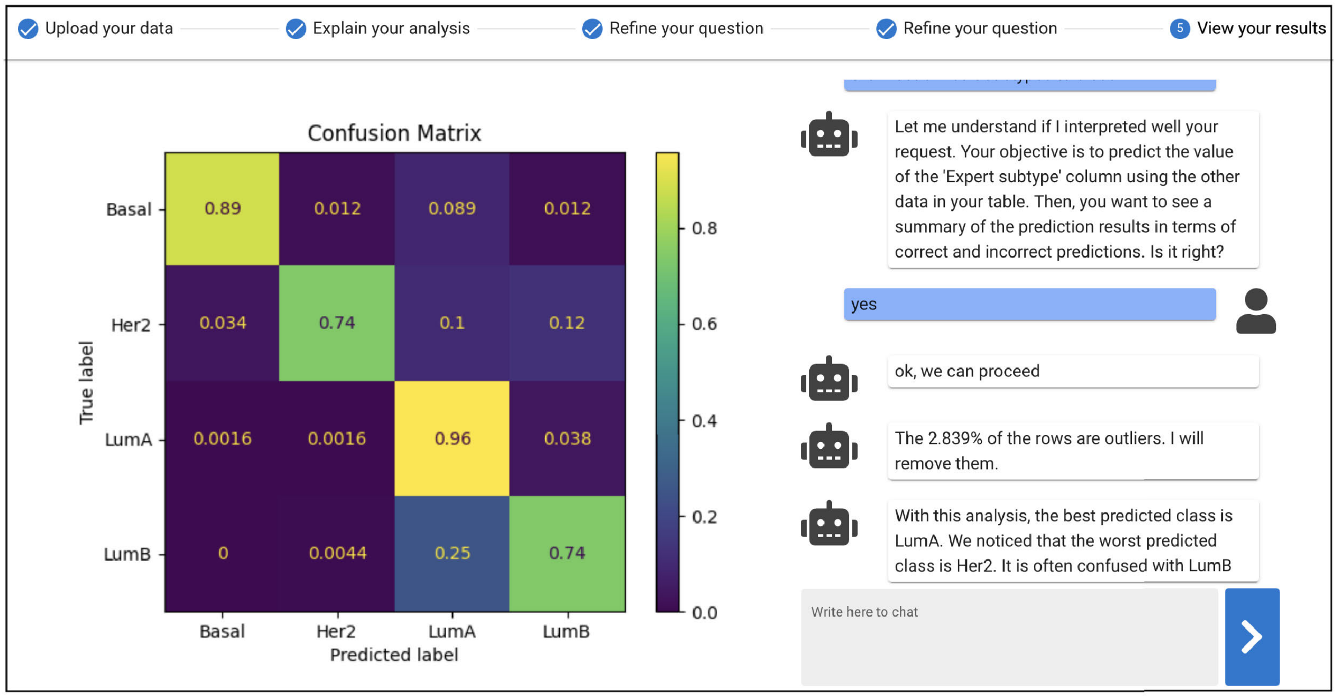
We considered a scenario in which the user does not specify the algorithm to be used for the classification or regression task but lets DSBot automatically select one algorithm and tune it hyper-parameters.

The evaluation was performed on datasets selected among the ones on which TPOT was evaluated by its authors[1] and some from Kaggle[2]. The final set comprises a total of 18 datasets for classification and 12 for regression. In addition, we selected the collection of datasets to be as heterogeneous as possible, including many different domains.

Note that while DSBot is an end-to-end tool, able to perform a complete analysis, including data cleaning, data preprocessing and result visualization, TPOT only analyzes datasets with no missing values or categorical features (which have to be encoded in advance) in order to determine either the best classifier or the best regressor.

To allow the comparison, we fed DSBot with the original datasets; for TPOT, we filled the missing values using an

[1]http://www.randalolson.com/data/benchmarks/
[2]https://www.kaggle.com/datasets



Let me understand if I interpreted well your request. Your objective is to predict the value of the 'Expert subtype' column using the other data in your table. Then, you want to see a summary of the prediction results in terms of correct and incorrect predictions. Is it right?

yes

ok, we can proceed

The 2.839% of the rows are outliers. I will remove them.

With this analysis, the best predicted class is LumA. We noticed that the worst predicted class is Her2. It is often confused with LumB

FIGURE 7. Web interface for visualization of final results and chatbot explanations (use case A).

Iterative Imputer method and we encoded the categorical variables using a *one-hot-encoder*. Then, for each dataset, we performed the following workflow for 50 times and averaged the results:

- We randomly selected the 20% of the samples from the dataset, used those as held-out dataset and the remaining as training set;
- We run TPOT on the training set using 5 generations of populations of 50 pipelines; a typical TPOT pipelines may include feature selection, feature engineering, model selection, and parameter tuning.
- We run DSBot on the training set. It automatically builds a pipeline covering from the data preprocessing to the data visualization; as classifier (regressor) we used the `autoClassification` (`autoRegression`) module. We stopped the pipeline after the selection of the model, as we were not interested in the result presentation. Typical DSBot pipeline may include different method to handle missing values (impute, remove), encoding of categorical features, outlier removal ad feature selection.
- The cases have been carefully selected so that user intervention is unnecessary.
- Both the methods returns a classifier (regressor) pipeline. First of all we saved the time needed by the two systems to produce their candidate models.
- We applied the two candidate models on the held-out dataset and measure the accuracy for the classification tasks and the Root Mean Squared Error (RMSE) for the regression tasks.

Aggregated results in term of performances and execution times are reported in Tables 4 and 5. In both tables, each row corresponds to a dataset on which the pipeline was executed both with TPOT and DSBot. The columns contain the dataset dimensions (rows $\times$ columns), the average performance relative to 20 runs with DSBot with its standard deviation (accuracy for classification, RMSE for regression), the performance comparison between DSBot and TPOT (DSBot mean performance $-$ TPOT mean performance), the average time taken to execute one run with DSBot, the time comparison of execution time between DSBot and TPOT (DSBot mean time/TPOT mean time $\times$ 100).

Regarding the classification, DSBot obtained better performances in term of accuracy in 11 over 18 datasets: *vowel, vehicle, diabetes types, cleveland nominal, vote, chess, stroke, australian, dna, dermatology* and *ann thyroid*. In six of these cases, we obtained an accuracy greater than 95 %. Others four obtained an accuracy between 75% and 85%, while only *cleveland nominal* obtained a low accuracy equal to 56%. When DSBot had worst performances than TPOT, it always got a comparable accuracy (within 95% of TPOT's accuracy).

Also the execution time is shorter w.r.t. TPOT: for classification tasks on average DSBot spent 14.85% of the time required by TPOT for the same analysis, while for the regression it spent 3.46% of the time required by TPOT.

Regarding the regression, we obtained better performances on 8 over 12 datasets. In these cases, DSBot performs better than TPOT obtaining a smaller RMSE in much less time.

B. EVALUATION OF THE TRANSLATION INTO EXECUTABLE PIPELINE

To evaluate the performance of the GPT-2-based translator, we fine-tuned the model on the 99% of the corpus of

TABLE 4. Evaluation of classification tests.

Dataset	rows × columns	Mean (std) accuracy	△ mean acc.	time [*sec*]	% time TPOT
Yeast	1479× 8	0.5939 (3.91e-02)	-0.0134	3.17	1.77
Vowel	990× 13	0.9803 (1.00e-02)	**0.0657**	2.83	1.08
Vehicle	846× 18	0.8055 (3.67e-02)	**0.1432**	3.98	3.49
Breast cancer	286× 9	0.7008(5.34e-02)	-0.0292	1.05	3.02
Diabetes types	768× 8	0.7662 (3.91e-02)	**0.0341**	2.08	4.46
Cleveland nom.	303× 7	0.5688 (5.48e-02)	**0.0155**	1.51	3.42
Balance scale	625× 4	0.8904 (2.45e-02)	-0.0172	2.51	4.72
Vote	435× 16	0.9557 (2.36e-02)	**0.0086**	1.86	4.56
Chess	3196× 36	0.9860 (5.67e-03)	**0.0011**	35.21	17.94
Stroke	5110× 11	0.9500 (8.08e-03)	**0.0023**	223.38	131.11
Australian	690× 14	0.8500 (4.40e-02)	**0.0094**	3.23	6.44
Ecoli	327× 7	0.8530 (4.04e-02)	-0.068	1.50	3.24
Car evaluation	1728× 21	0.9528 (2.18e-02)	-0.0179	4.48	2.31
DNA	3186× 180	0.9498 (8.09e-03)	**0.0091**	63.91	5.89
Diabetes	768× 8	0.7701 (2.79e-02)	**0.0406**	2.53	5.43
Dermatology	366× 34	0.9479 (2.42e-02)	-0.0169	2.02	2.79
Adult	48842× 14	0.8661 (2.20e-03)	-0.0042	669.24	53.14
Ann thyroid	7200× 21	0.9962 (1.42e-03)	**0.0099**	59.56	12.41

TABLE 5. Evaluation of regression tests.

Dataset	rows × columns	Mean (std) RMSE	△ mean RMSE	Time [*sec*]	% Time TPOT
Sample regression ds	10000× 21	1.40e-01 (8.92e-02)	4.51e-02	116.77	11.89
Students performance	1000× 8	2.69e-09 (5.99e-11)	**-1.71e-11**	1.62	3.31
House price	545× 12	1.11e+06 (7.32e+04)	**-1.14e+05**	1.56	3.43
Real estate	414× 7	8.09e+00 (1.81e+00)	4.82e-01	1.04	1.80
Material strength	1030× 8	5.54e+00 (8.96e-01)	3.06e-01	2.24	3.62
Patients LOS	835× 4	2.51e+02 (1.14e+02)	1.49e+01	2.35	2.54
Possum length	104× 13	2.08e+00 (2.29e-01)	**-7.79e-02**	0.53	1.41
Insurance price	1338× 6	4.60e+03 (3.99e+02)	**-8.1e+02**	2.10	2.50
Boston houses	506× 13	3.48e+00 (7.30e-01)	**-8.67e-02**	1.42	2.78
Startup marketing	50× 4	9.02e+03 (2.76e+03)	**-2.54e+02**	0.29	1.13
Insurance expenses	1338× 6	4.62e+03 (4.70e+02)	**-8.52e+01**	2.13	3.31
Second hand cars	1000× 11	8.90e+03 (4.40e+02)	**-1.22e+02**	2.01	3.85

sentences, to test it on the remaining 1% of the sentences in the dataset for validation.

We used the BLEU score [42], a widely-used metric in machine translation, to measure the performance of our model. In our experiment, we computed the score up to 4-grams as our DAW sentences range between 1 and 4 tokens. The BLEU score ranges from 0 to 1 and reflects the similarity of the machine-translated text to a set of high-quality reference translations [42].

In our case, we confronted the translation produced by GPT-2 with a set of possible DAW sentences for every element of the test. This set was created by adding sentences obtained by replacing low-level DAW terms with their corresponding high-level ones. For example, if the DAW translation of a sentence was `outliersRemove randomForest`, its variations were `outliersRemove classification`, and `outliers randomForest`.

With this metric, GPT-2 was able to achieve a score of 0.8, indicating that the model's translations were highly similar to the reference translations.

Overall, our use of GPT-2 for machine translation from English to DAW proved to be an effective approach. The model's ability to handle input sequences and understand context, combined with its high level of adaptability, made it an ideal choice for this task. Furthermore, the high BLEU score achieved by the model demonstrates its ability to generate accurate translations.

VI. CONCLUSION

This paper describes DSBot, a novel approach and system to build and execute data analysis pipelines starting from natural language requests and datasets uploaded by the user. The most significant aspects of DSBot include the definition of DAW, a domain specific language for describing data analysis pipelines; a machine translation based on a neural network for producing a DAW sequence from the user's query; a matching algorithm to extract the best matching pipeline out of a dictionary of pipelines; and a conversational agent interacting with the user whenever necessary.

DSBot does not merely execute the operations explicitly requested by the user, but automatically augments the pipeline so as to improve the result, in a way that is accessible to a user who is not deeply expert in data analysis. We provide empirical evidence of the potentiality of such a tool by discussing its evolution in two case studies: one to investigate the advantages of the conversation, the other in which DSBot finds the optimal pipeline to answer the user's research question.

DSBot is a first step towards the exploitation of Data Science by non-experts, paving the ground for a new family of tools that makes Data Science more accessible and usable by a larger audience.

Still, there are some open issues to address.

In the current implementation, no actions are taken in the case of an unbalanced dataset. This problem may be addressed by either adopting rebalancing strategies during the data preprocessing steps (e.g., downsampling or oversampling) or by computing different evaluation metrics (e.g., Matthew Correlation Coefficient). An additional area of improvement is the automatic selection of machine learning models, which in the current version of the system is based on Accuracy metric, which is not appropriate in situations of unbalanced data sets and, in such situations should be substituted by more informative metrics (e.g. balanced accuracy). Given the modular nature of the DSBot, these extensions will be relatively easy to develop and will be achieved in the near future.

Other improvements concern two main issues: i) widening and improving the set of pipelines supported by DSBot and the operations supported by system that are currently allowed on single-table data only; ii) enhancing the conversational power of the chatbot in several directions: to elicit a wider number of user's research questions, to sustain a more natural interaction, and to increase the transparency of the AutoML processes and the explainability of the system (defined in [43] as the degree at which an AI system can ''enable human users to understand, appropriately, trust, and effectively manage artificially intelligent partners'').

Our plan is to address the first issue not only by integrating and automatizing more algorithms and analysis modules, e.g., methods used for survival analysis or time series, but also by testing them on wide number of datasets, to enrich the number of supported pipelines, refine their quality, and provide more advanced computational support to the end user. Concerning the chatbot, we will extend the training corpora for research question interpretation and elicitation by collecting many new real-world research questions, with alternative formulations, and new exemplary conversation flows. We will investigate the existing techniques of explainable AI [43] to identify those more appropriate for the AutoML approach of DSBot. They will inform the design of new conversational patterns for the chatbot [44], to provide the user with explanations that make the analysis processes and their outcomes more transparent, comprehensible, and trustful.

APPENDIX A

Operations supported by DAW - Domain Specific Language for building data analysis workflows

- **missingValuesHandle:** first removes columns that contain more than 50% of missing values, then computes the percentage of rows with missing values: if they are less than 5%, then missing values are removed; if they are between 5% and 10%, then missing values are filled; otherwise, the user is asked to check these rows and make decisions about them.
- **fillMissingValues:** using iterative imputation, fills the numerical missing values; while the categorical ones are filled with the most common value in the column.
- **removeMissingValues:** removes the rows with missing values.
- **oneHotEncoder:** applies one hot encoding to categorical columns so that the dataset includes only either numerical values or 0/1 values.
- **outliersRemove:** removes the rows that have more than 90% of elements that are considered outliers, i.e., whose difference between the element value and the mean value exceeds the triple of the standard deviation.
- **zeroVarianceRemove:** removes the columns that have a variance equal to zero.
- **correlatedFeaturesRemove:** considering two features that have a correlation higher than 0.9, it considers only one of those two features and drop the other.
- **removeFeatures:** asks the users if they want to remove the features that include more than 50% of different elements.
- **standardization:** returns a standardized dataset by applying the Standard Scaler method.
- **normalization:** returns a normalized dataset by applying a min-max scaler method.
- **labelRemove:** drops the target column from the dataset.
- **labelAppend:** re-appends the target column to the dataset, when needed.
- **pearson:** computes the correlation matrix by applying Pearson measure.
- **spearman:** computes the correlation matrix by applying Spearman measure.
- **autoClassification:** tries different combinations of parameters and classification modules among the available ones in order to retrieve the method along with the parameters yielding to the best accuracy score.
- **randomForestClassifier:** implements the random forest classifier with the tuning of the parameters.
- **logisticRegression:** implements the logistic regression classifier with the tuning of the parameters.
- **kNeighborsClassifier:** implements the kNeighbors classifier with the tuning of the parameters.
- **adaBoostClassifier:** implements the adaBoost classifier with the tuning of the parameters.
- **autoRegression:** tries different combinations of parameters and regression modules among the available ones in order to retrieve the method along with the parameters that are able to obtain the best mean squared error.
- **linearRegression:** implements the linear regression with the tuning of the parameters.
- **randomForestRegressor:** implements the random forest regressor with the tuning of the parameters.
- **ridgeRegression:** implements the ridge regression with the tuning of the parameters.
- **adaBoostRegressor:** implements the adaBoost regressor with the tuning of the parameters.
- **kmeans:** implements the kmeans algorithm for clustering with the tuning of the parameters by applying Grid Search.
- **dbscan:** implements the dbscan algorithm for clustering with the tuning of the parameters by applying Grid Search.

- **agglomerativeClustering:** implements agglomerative clustering with the tuning of the parameters by applying Grid Search.
- **lasso:** applies Lasso linear model with iterative fitting along a regularization path to perform feature selection.
- **selectKBest:** selects features according to the k highest scores.
- **laplace:** performs feature selection with an unsupervised method that uses the Laplacian score and selects the features with the highest one.
- **userFeatureSelection:** allows the user to decide which features to keep and which to remove.
- **featureImportance:** retrieves the importance of the features after a classification or a regression algorithm.
- **pca2:** performs a principal component analysis, creating a 2D representation of the dataset (linear reduction).
- **mds2:** performs multi-dimensional scaling, creating a 2D representation of the dataset (non-linear reduction).
- **apriori:** implements the apriori algorihtm in order to retrieve the frequent itemsets and the association rules that lay in the provided data.
- **regressionPerformance:** after a regression algorithm. it computes performance measures, such as r2, mean squared error, root mean squared error and mean absolute error.
- **confusionMatrix:** produces a summary of prediction results on a classification problem.
- **scatterplot:** produces a scatterplot of the data, e.g., after a clustering problem.
- **clustermap:** supplies a clustermap of the correlation matrix.
- **roc:** computes the ROC curve and the area under the curve after having applied a classification method.
- **lassoPlot:** provides a barplot of the features extracted with lasso.
- **tableRegression:** provides the regression performances in a table to show to the user.
- **tableAssociationRules:** saves the association rules to be presented to the user.
- **featureImportancePlot:** shows in a pie chart the importance of the features after a classification or regression problem.

ACKNOWLEDGMENT

(Sara Pidó and Pietro Pinoli are co-first authors.)

REFERENCES

[1] K. Witkowski, "Internet of Things, big data, industry 4.0—Innovative solutions in logistics and supply chains management," *Proc. Eng.*, vol. 182, pp. 763–769, Jan. 2017.

[2] F. Murtagh and K. Devlin, "The development of data science: Implications for education, employment, research, and the data revolution for sustainable development," *Big Data Cognit. Comput.*, vol. 2, no. 2, p. 14, Jun. 2018.

[3] J. Honaker and V. D. Orazio. *Statistical Modeling by Gesture: A Graphical, Browser-Based Statistical Interface for Data Repositories*. Accessed: Dec. 31, 2021. [Online]. Available: http://ceur-ws.org/Vol-1210/datawiz201405.pdf

[4] C. V. K. Iyer, F. Hou, H. Wang, Y. Wang, K. Oh, S. Ganguli, and V. Pandey, "Trinity: A no-code AI platform for complex spatial datasets," in *Proc. 4th ACM SIGSPATIAL Int. Workshop AI Geograph. Knowl. Discovery*, New York, NY, USA, Nov. 2021, pp. 33–42.

[5] P. Crovari, S. Pidò, P. Pinoli, A. Bernasconi, A. Canakoglu, F. Garzotto, and S. Ceri, "GeCoAgent: A conversational agent for empowering genomic data extraction and analysis," *ACM Trans. Comput. Healthcare*, vol. 3, no. 1, pp. 1–29, Jan. 2022.

[6] R. S. Olson and J. H. Moore, "TPOT: A tree-based pipeline optimization tool for automating machine learning," in *Proc. Workshop Autom. Mach. Learn.*, 2016, pp. 66–74.

[7] V. G. Renumol, D. Janakiram, and S. Jayaprakash, "Identification of cognitive processes of effective and ineffective students during computer programming," *ACM Trans. Comput. Educ.*, vol. 10, no. 3, pp. 1–21, Aug. 2010.

[8] D. Price, E. Rilofff, J. Zachary, and B. Harvey, "NaturalJava: A natural language interface for programming in Java," in *Proc. 5th Int. Conf. Intell. User Interfaces*, 2000, pp. 207–211.

[9] M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, "A survey of machine learning for big code and naturalness," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–37, 2018.

[10] J. Shin and J. Nam, "A survey of automatic code generation from natural language," *J. Inf. Process. Syst.*, vol. 17, no. 3, pp. 1–10, Jun. 2021.

[11] S. Chong and R. Pucella, "A framework for creating natural language user interfaces for action-based applications," 2004, *arXiv:cs/0412065*.

[12] D. Vadas and J. R. Curran, "Programming with unrestricted natural language," in *Proc. Australas. Lang. Technol. Workshop*, 2005, pp. 191–199.

[13] P. Yin and G. Neubig, "A syntactic neural model for general-purpose code generation," 2017, *arXiv:1704.01696*.

[14] H. Lieberman and M. Ahmad, "Knowing what you're talking about: Natural language programming of a multi-player online game," in *No Code Required*. Amsterdam, The Netherlands: Elsevier, 2010, pp. 331–343.

[15] V. Le, S. Gulwani, and Z. Su, "SmartSynth: Synthesizing smartphone automation scripts from natural language," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2013, pp. 193–206.

[16] X. V. Lin, C. Wang, D. Pang, K. Vu, and M. D. Ernst, "Program synthesis from natural language using recurrent neural networks," Univ. Washington Dept. Comput. Sci. Eng., Seattle, WA, USA, Tech. Rep., UW-CSE-17-03-01, 2017.

[17] J. Lin, Y. Liu, Q. Zeng, M. Jiang, and J. Cleland-Huang, "Traceability transformed: Generating more accurate links with pre-trained BERT models," in *Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng. (ICSE)*, May 2021, pp. 324–335.

[18] M. Chen et al., "Evaluating large language models trained on code," 2021, *arXiv:2107.03374*.

[19] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowl.-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106622.

[20] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2013, pp. 847–855.

[21] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Auto-sklearn: Efficient and robust automated machine learning," in *Automated Machine Learning*. Cham, Switzerland: Springer, 2019, pp. 113–134.

[22] S. K. Karmaker, M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni, "AutoML to date and beyond: Challenges and opportunities," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–36, Nov. 2022.

[23] T. De Bie, L. De Raedt, J. Hernández-Orallo, H. H. Hoos, P. Smyth, and C. K. I. Williams, "Automating data science: Prospects and challenges," 2021, *arXiv:2105.05699*.

[24] D. Bouneffouf, C. Aggarwal, T. Hoang, U. Khurana, H. Samulowitz, B. Buesser, S. Liu, T. Pedapati, P. Ram, A. Rawat, M. Wistuba, and A. Gray, "Survey on automated end-to-end data science?" in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–9.

[25] E. Ozan, "A novel browser-based no-code machine learning application development tool," in *Proc. IEEE World AI IoT Congr. (AIIoT)*, May 2021, pp. 282–284.

[26] D. Smilkov, N. Thorat, Y. Assogba, A. Yuan, N. Kreeger, P. Yu, K. Zhang, S. Cai, E. Nielsen, D. Soergel, S. Bileschi, M. Terry, C. Nicholson, S. N. Gupta, S. Sirajuddin, D. Sculley, R. Monga, G. Corrado, F. B. Viégas, and M. Wattenberg, "TensorFlow.Js: Machine learning for the web and beyond," 2019, *arXiv:1901.05350*.

[27] M. Carney, B. Webster, I. Alvarado, K. Phillips, N. Howell, J. Griffith, J. Jongejan, A. Pitaru, and A. Chen, ''Teachable machine: Approachable web-based tool for exploring machine learning classification,'' in *Proc. Extended Abstr. CHI Conf. Hum. Factors Comput. Syst.*, Apr. 2020, pp. 1–8.

[28] S. Narkar, Y. Zhang, Q. V. Liao, D. Wang, and J. D. Weisz, ''Model LineUpper: Supporting interactive model comparison at multiple levels for AutoML,'' in *Proc. 26th Int. Conf. Intell. User Interfaces*, Apr. 2021, pp. 170–174.

[29] R. Chang, ''Snowcat and CAVA: Visualization tools for interacting with AutoML and knowledgebases,'' Tufts Univ., Medford, MA, USA, Tech. Rep. AD11479, 2021.

[30] D. Wang, J. Andres, J. D. Weisz, E. Oduor, and C. Dugan, ''AutoDS: Towards human-centered automation of data science,'' in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, New York, NY, USA, May 2021, pp. 1–12.

[31] L. Meng, S. van den Elzen, and A. Vilanova, ''ModelWise: Interactive model comparison for model diagnosis, improvement and selection,'' *Comput. Graph. Forum*, vol. 41, no. 3, pp. 97–108, Jun. 2022.

[32] G. King, ''An introduction to the dataverse network as an infrastructure for data sharing,'' *Sociol. Methods Res.*, vol. 36, no. 2, pp. 173–199, Nov. 2007.

[33] A. S. Maiya, ''Ktrain: A low-code library for augmented machine learning,'' 2020, *arXiv:2004.10703*.

[34] R. J. L. John, N. Potti, and J. M. Patel, ''AVA: From data to insights through conversations,'' in *Proc. CIDR*, 2017, pp. 1–10.

[35] E. Fast, B. Chen, J. Mendelsohn, J. Bassen, and M. S. Bernstein, ''IRIS: A conversational agent for complex tasks,'' in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, New York, NY, USA, Apr. 2018, pp. 1–12.

[36] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman, ''Metrics for explainable AI: Challenges and prospects,'' 2018, *arXiv:1812.04608*.

[37] V. Likic, ''The Needleman-Wunsch algorithm for sequence alignment,'' Lect. Given 7th Melbourne Bioinf. Course, Bi021 Mol. Sci. Biotechnol. Inst., Univ. Melbourne, Melbourne, VIC, Australia, 2008, pp. 1–46.

[38] Seaborn. *Penguins.csv*. Accessed: Apr. 28, 2022. [Online]. Available: https://github.com/mwaskom/seaborn-data/blob/master/penguins.csv

[39] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, ''RASA: Open source language understanding and dialogue management,'' 2017, *arXiv:1712.05181*.

[40] X. Dai, T. Li, Z. Bai, Y. Yang, X. Liu, J. Zhan, and B. Shi, ''Breast cancer intrinsic subtype classification, clinical use and future trends,'' *Amer. J. Cancer Res.*, vol. 5, no. 10, p. 2929, 2015.

[41] Kaggle. *Stroke Prediction Dataset*. Accessed: Apr. 28, 2022. [Online]. Available: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

[42] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, ''BLEU: A method for automatic evaluation of machine translation,'' in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2001, pp. 311–318.

[43] A. Adadi and M. Berrada, ''Peeking inside the black-box: A survey on explainable artificial intelligence (XAI),'' *IEEE Access*, vol. 6, pp. 52138–52160, 2018.

[44] Q. V. Liao, D. Gruen, and S. Miller, ''Questioning the AI: Informing design practices for explainable AI user experiences,'' in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, Apr. 2020, pp. 1–15.

SARA PIDÓ received the bachelor's and master's degrees in computer science and engineering from Politecnico di Milano, Italy, where she is currently pursuing the Ph.D. degree in data analytics and decision sciences. She is a member of the DEIB Bioinformatics Group. Her research interests include the application of data science methods and algorithms to bioinformatics, with particular attention to the design and implementation of conversational agents to support bioinformatics analysis.

PIETRO PINOLI received the Ph.D. degree (cum laude), in 2017. His Ph.D. thesis was titled ''Modeling and Querying Genomic Data,'' where he proposed and benchmarked data structures and algorithms to manage, search and elaborate huge collections of genomic datasets, by means of cloud and distributed technologies. He was a Visiting Ph.D. Student with Harvard University, Cambridge, MA, USA. He was a Researcher Fellow and a Lecturer with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, Italy. He participated in the Italian PRIN GenData, ERC GeCo, and EIT VirusLab projects. His research interests include bioinformatics and computational biology, databases and data management, big data technology and algorithms, machine learning, natural language processing, and drug repurposing.

PIETRO CROVARI received the bachelor's degree in computer science and engineering from the University of Genoa, Italy, and the M.S. degree in computer science and engineering from Politecnico di Milano, where he is currently pursuing the Ph.D. degree in information technology. His research interests include the design and implementation of multimodal conversational interfaces for process-intensive applications, particularly in the domain of data science and bioinformatics research.

FRANCESCA IEVA was born in Milan, Italy, in 1984. She received the M.S. degree in mathematical engineering, in 2008, and the Ph.D. degree in mathematical models and methods for engineering, in 2012. She is currently an Associate Professor of statistics with the Department of Mathematics, Politecnico di Milano. She is also the Associate Head of the Center for Health Data Science, Human Technopole. She is the principal investigator of three national project grants and a number of funded projects. She is also on the Advisory Board of the Center for Healthcare Research and Pharmacoepidemiology. Her mentoring activity is comprehensive of more than 20 M.D. students and six Ph.D. students. Her research interests include health analytics and statistical learning in a biomedical context. She is a member of the Italian Statistical Society and the International Society of Clinical Biostatistics. She is an Associate Editor of *Statistical Methods and Applications*.

FRANCA GARZOTTO received the bachelor's and master's degrees in mathematics from the University of Padua, Italy, and the Ph.D. degree in information engineering from Politecnico di Milano, Italy. She is currently an Associate Professor with Politecnico di Milano. She is also the Director of i3Lab (https://i3lab.polimi.it/). I3Lab is a multidisciplinary research laboratory focusing on advanced interactive technologies. Her main research interests include conversational agents, embodied systems, and mixed reality environments.

STEFANO CERI (Member, IEEE) is currently a Professor with Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano. He received two advanced ERC grants, on search computing and data-driven genomic computing (GeCo, 2016–2021). He has authored more than 350 publications and ten international books. His research interests include extending database technology and applying them as a data scientist, with a recent emphasis on genomics and viruses. He is a fellow of ACM. He received the ACM-SIGMOD Innovation Award.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement