# DORIAN in action: Assisted Design of Data Science Pipelines

Sergey Redyuk
TU Berlin
sergey.redyuk@tu-berlin.de

Zoi Kaoudi
TU Berlin
zoi.kaoudi@tu-berlin.de

Sebastian Schelter
University of Amsterdam
s.schelter@uva.nl

Volker Markl
TU Berlin
volker.markl@tu-berlin.de

## ABSTRACT

Existing automated machine learning solutions and intelligent discovery assistants are popular tools that facilitate the end-user with the design of data science (DS) pipelines. However, they yield limited applicability for a wide range of real-world use cases and application domains due to (a) the limited support of DS tasks; (b) a small, static set of available operators; and (c) restriction to evaluation processes with quantifiable loss functions. We demonstrate DORIAN, a human-in-the-loop approach for the *assisted design of data science pipelines* that supports a large and growing set of DS tasks, operators, and arbitrary user-defined evaluation processes. Based on the user query, i.e., a dataset and a DS task, DORIAN computes a ranked list of candidate pipelines that the end-user can choose from, alter, execute and evaluate. It stores executed pipelines in an experiment database and utilizes similarity-based search to identify relevant previously-run pipelines from the experiment database. DORIAN also takes user interaction into account to improve suggestions over time. We show how users can interact with DORIAN to create and compare DS pipelines on various real-world DS tasks without the need for writing any code.

**PVLDB Reference Format:**
Sergey Redyuk, Zoi Kaoudi, Sebastian Schelter, and Volker Markl. DORIAN in action: Assisted Design of Data Science Pipelines. PVLDB, 15(12): 3714 - 3717, 2022.
doi:10.14778/3554821.3554882

## 1 INTRODUCTION

To facilitate users with the overwhelming task of designing data science (DS) pipelines, automated machine learning (AutoML) solutions [6, 12] are commonly used in specific tasks, such as supervised classification. They usually navigate a search space of ML models and hyperparameters and execute these pipelines to determine the best one. Another approach that provides guidance for pipeline design includes intelligent discovery assistants (IDA). These focus on scenarios where end-users are kept in-the-loop and utilize meta-features, knowledge- or case- bases [1, 3, 5, 8, 9, 11].

Both families of solutions, however, rarely address domain-specific tasks and the challenges faced in creative experimentation. First, certain domains introduce domain-specific techniques (e.g., marker gene identification in genomics and single-cell analysis) which are not readily available in existing AutoML and IDA tools. Second, modern DS pipelines can get extremely complex, containing a wide range of operators from different domains. If current AutoML tools were to support all these operators, their search space would explode requiring extensive resources and time to find optimal pipelines. Third, it is prevalent in many applications that domain experts cannot quantify or automate the pipeline evaluation, and hence, the selection among pipeline candidates. Instead, they manually inspect the results of different pipeline variants, compare them, and decide which one is best. Such an evaluation process does not include a well-specified, quantifiable metric (e.g., AUC ROC score) and thus, is out of scope for AutoML and IDA solutions.

We address the aforementioned challenges and demonstrate DORIAN[1], a human-in-the-loop approach for the assisted design of DS pipelines. Our goal is the support of a broad range of application domains (i.e., arbitrary DS tasks, operators, and evaluation processes) that go beyond the state-of-the-art solutions in three aspects: (a) user-input is essential and fully automated solutions [4, 10] might be inapplicable; (b) a small, static set of supported operators and common "best practices" are not powerful enough for particular domain-specific tasks; and (c) adaptation to new operators is performed fast, with little to no overhead for the end-user.

Given a dataset and a DS task (e.g., classification), DORIAN computes a ranked list of pipeline candidates that the end-user can choose to execute or modify. To achieve this, it uses a dynamically updated experiment database to store all previously executed experiments from different teams and domains. DORIAN is also composed of a recommendation engine that utilizes the experiment database and incorporates user interactions to improve individual suggestions over time. The benefits of DORIAN are numerous: (a) The recommendation engine is *online* (i.e., it does not have delays in the incorporation of new user input) and *extensible* by the end-users: When an end-user adds a new operator or a DS task, it becomes available to other users immediately after the validation. (b) DORIAN treats pipeline suggestion as a search problem as opposed to the expensive "generate-train-evaluate" loop of AutoML solutions. This allows us to benefit from the advancements in the data management field and employ efficient search over data, pipelines, and previous user interaction. (c) It brings reasonable explainability to the pipeline selection process via its ranking.

In this demo, we showcase DORIAN via an intuitive and easy-to-use graphical user interface (GUI). Attendees will be able to design

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554882

[1]The original acronym stands for a tool for **R**eproducibity, **I**nspection, and **A**utomatio**N** of **D**ata-**O**riented experiments.

pipelines for a set of 11 predefined real-world tasks and datasets. In particular, they will be able to: (i) choose among different pipelines, (ii) modify existing pipelines, (iii) execute pipelines, and (iv) visually inspect and compare pairs of pipelines. The purpose of the demo is twofold: first, to highlight that novice users can effortlessly construct DS pipelines in a few steps with zero code, and second, to show that domain experts can easily plug their own operators and use their own objective criteria which are instantaneously picked up and used by the recommendation engine.

## 2 SYSTEM OVERVIEW

DORIAN tackles the problem of *assisted design of DS pipelines*: Given a dataset and a DS task, together with a user-defined evaluation process of choice (e.g., AUC ROC score or any custom-made evaluation) and a time budget, output the best set of pipeline candidates according to the evaluation process.

The core challenges that DORIAN tackles are: (i) how to effectively support a large and growing set of operators included in the suggested pipelines, DS tasks, and arbitrary evaluation processes that potentially lack quantifiable loss functions; and (ii) how to provide interactive suggestions to the end-user (a response under 1 second is considered real-time user interaction [7]) while incorporating new user input without delays. Figure 1 shows an overview of DORIAN. On the left side we illustrate the user interaction, while on the right side we see the internal of the system. The user interaction starts with a user query (i.e., a dataset, task and optionally evaluation process and predefined DS pipeline) ❶. DORIAN then outputs the first set of ranked pipeline candidates (initial recommendation) ❷. As part of the *User Interaction Cycle*, the user can (*i*) edit the chosen candidate by adding, removing, or altering operators in the pipeline; (*ii*) discard the candidate as irrelevant based on the domain knowledge or personal preference; (*iii*) submit the pipeline for execution to generate the output data. After users receive the results ❹, they can choose to modify their pipeline again or go back to another pipeline candidate and request new suggestions. This process is iterative. DORIAN can be populated with DS pipelines found in publicly available sources, such as Kaggle or OpenML ⓿.

To achieve this, DORIAN comprises four main components: the *DS Pipeline Extractor*, the *Experiment Database* (ExpDB), the *Recommendation Engine* and the *Execution engine* (Figure 1, gray boxes).

**DS Pipeline Extractor.** It parses the source code of a DS pipeline into an Abstract Syntax Tree (AST) and extracts a succinct but informative graph out of the AST. To achieve this, the *DS pipeline extractor* utilizes graph rewrite rules in addition to a knowledge graph. The latter adds semantics to the DS pipelines which is crucial for providing useful recommendations to the end-users.

**Experiment Database.** ExpDB acts as a centralized data management component for storing information from all previous DS experiments across teams and domains or collected from other publicly available experiment databases, e.g., OpenML and Kaggle. This component provides the following novel aspects: (*i*) an extensible intermediate representation for executed pipelines that allows for new user-defined operators and tasks, (*ii*) efficient search over the constantly growing database of past experiments that enables interactive suggestions, and (*iii*) storing a succinct representation of the datasets instead of the raw files. Information about the experiments

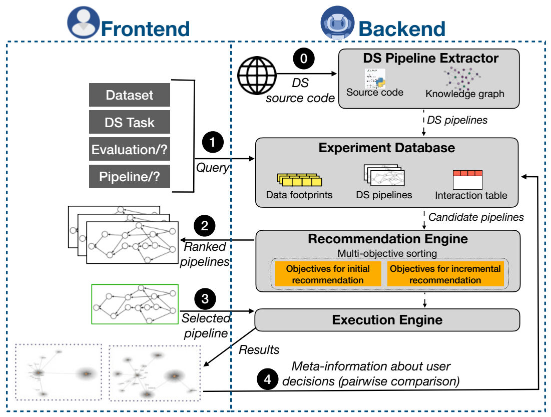

**Figure 1: Framework overview**

(e.g., dataset ID, pipeline ID, evaluation metric) and user interaction (e.g., selected, executed, discarded, compared pipeline candidates) is also stored in the *Interaction Table* inside the *Experiment Database*.

ExpDB is responsible for finding pipeline candidates and passing them to the recommendation engine. It does this in two ways: In the first interaction cycle, it finds pipelines that were previously executed on *datasets similar* to the input dataset. In the susequent interaction cycles and after the user has chosen at least one pipeline, it retrieves *pipelines similar* to the selected ones. In addition, ExpDB records all pipeline runs and user interactions to further improve its suggestions. It achieves these based on three main elements: (a) fixed-size numeric *data footprints* that contain a subset of meta-features from [4] extracted from the input dataset and used for the KDD-tree based similarity search of previously used datasets, (b) a graph-based representation of the *data science pipelines* that allows to use the graph-edit distance as a discrete similarity measure for similarity search based on the BK-trees, and (c) the relational representation of *user interactions* that is indexed for efficient selection.

**Recommendation Engine.** It selects and ranks pipeline candidates for a given user query *in real-time* and updates suggestions as new information regarding the user decisions (i.e., choices) becomes available. We distinguish two types of suggestions: the initial recommendation and the suggestions for incremental improvements. As both types of suggestions exhibit multiple requirements for ranking, we frame the recommendation of pipelines with the problem of multi-objective sorting. The *Recommendation Engine* first receives from ExpDB a set of pipeline candidates that were previously executed on a similar dataset and filters out the ones that do not perform the user-specified DS task $T$. It ranks the candidates based on relevance to the user query and the performance metric (if available), giving preference to high variety of suggested candidates as well as to previous user interaction. To specify the "relevance", the *Recommendation Engine* uses a set of default ranking objectives, such as predictive performance, similarity of pipelines, etc. Pipeline candidates are then ordered based on the specified list of ranking objectives (e.g., "unseen pipelines first", "pipelines with higher average predictive performance first", "pipelines with higher performance on similar data first") that is exposed to the end-user and can be tailored to each user query and evolve over time.

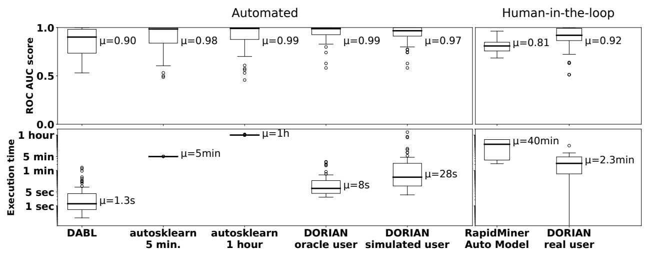

**Figure 2: Evaluation of DORIAN against six baselines**

**Execution Engine.** It is an abstraction of the computational environment. In this demonstration, we use the `Python3.9` runtime with the common data science toolkit installed (the `sklearn`, `pandas`, `keras` libraries). It temporarily stores the results of pipeline candidates that the end-user chose to execute in order to support their pairwise comparison in retrospective.

Note that the evaluation processes are not limited to standard ML performance metrics such as accuracy or ROC AUC score. In the case when well-specified quantifiable evaluation metrics do not exist, the end-user can manually compare the output data of any given pair of executed candidates in order to identify a "better" candidate w.r.t. the chosen evaluation process. Recording the end-user decisions regarding the pairwise candidate comparison in the *Experiment Database* allows for improving the ranking of the pipeline candidates over time. `DORIAN` suggests new candidates until the user finds a suitable pipeline.

**Results.** To evaluate the quality of `DORIAN`'s pipeline recommendation, we use 63 verified datasets from the `OpenML-CC18` benchmark suite [2] as user queries and 3900 `OpenML` pipelines to populate the experiment database. We apply a leave-one-out scheme where the experiments related to all but one datasets are kept in the ExpDB, 1 dataset with the underlying experiments specifies a previously unseen user query. As baselines, we use: (i) the `DABL`[2] library for baseline pipeline generation that applies a static set of pipeline synthesis rules, (ii) `auto-sklearn` [4] as an AutoML solution using two variants: one with total execution budget of 5min and another with the total execution budget set to 1h, (iii) `RapidMiner Auto Model`[3] as an IDA solution, and (iv) a synthetic "oracle" baseline that imitates end-users that know the optimal path to the best performing pipeline candidate. For `DORIAN`, we run two variants - one with real users as part of a preliminary user study and another with simulated user behavior that is modeled analogous to the breadth-first search: all candidates that are suggested during the first iteration are selected and executed in the direct order without discarding or skipping, and then their incremental updates are selected and executed in the direct order as well.

Figure 2 depicts the results. We report the average predictive performance and the total execution time (human-in-the-loop baselines include the delays for user interaction). `DORIAN` *performs similarly to its automated counterparts while being faster and extensible.* `DORIAN` with a simulated user represents a trade-off that performs better than the simple baseline and reaches predictive performance comparable to the AutoML solutions fast. On average, the total execution time of `DORIAN` is one order of magnitude faster than the automated `auto-sklearn 5min` and `auto-sklearn 1h` baselines.

`DORIAN` (operated by a real user) exceeds the average performance of `RapidMiner` and reaches predictive performance close to that of automated tools, i.e., its ROC AUC is 0.92 while `auto-sklearn 5 min` scored 0.95. `RapidMiner` underperforms due to the fixed set of applied ML models and their hyperparameters, and the lack of preprocessing mechanisms except for feature selection. The total time of our approach includes many iterations of suggestions, where a single iteration takes under 0.5s (and the computation of data footprints take under 3s) which adheres to the standards of interactive user experience. Importantly, `DORIAN` works under relaxed assumptions that do not restrict the support of any DS tasks or operators compared to other baselines.

## 3 DEMONSTRATION

In the following, we describe the core user interaction scenario and the functionality of the demonstrated user interface (UI, Figure 3). The audience will have a set of eleven real datasets and tasks to choose from. The domains of the data range from thyroid disease prediction to the analysis of internet advertisements. All available datasets fit the demonstration scenario presented below.

**Scenario.** We imagine *Alice* who is a domain expert without formal education in data science. They work on the topic of biological response prediction from the chemical properties of molecules and require a pipeline to predict the biological response on implantation of a medical device based on the chemical structure of the materials used for manufacturing. The data at hand represent 1776 molecular descriptors (i.e., calculated properties that capture some of the characteristics of the molecule - for example size, shape, or elemental constitution) and a binary signal that describes if a particular molecule was seen to elicit a response.
The audience will be able to play a role of *Alice* and perform the following steps (see Figure 3): specify the user query ❶, review suggested pipelines ❷, select and alter the candidate of interest ❸, execute and compare the selected pipeline candidates pair-wise ❹.
❶ *User Query.* Initially, *Alice* specifies a query, i.e., uploads an input dataset and chooses the task from a drop-down list. Optionally, they provide as inputs: (*i*) the specification of a quantifiable performance evaluation process, if it exists (e.g., 5-fold cross-validation and AUC ROC score), and (*ii*) an initial DS pipeline to work with if they would like suggestions for improvements.
❷ *Candidate Review.* `DORIAN` provides a ranked list of pipelines sorted based on a default set of ranking objectives. Recommendations appear incrementally in several iterations, thus, the feed provides the capability to review the history of suggestions. The end-user can select one of the candidates for detailed review.
❸ *Candidate Selection and Alteration. Alice* selects a pipeline candidate for manual alteration and can use it as a source pipeline for incremental improvements or discard it. Specifically, they can modify the hyperparameters of the provided operators, replace any operator with an alternative of similar type, remove an operator, add an existing one or create their own by using the code editor. *Alice* can request new suggestions based on the selected candidate.

[2]https://amueller.github.io/dabl/dev/, DABL library, *accessed: 24.03.2022*
[3]https://docs.rapidminer.com/latest/studio/guided/auto-model/index.html, RapidMiner Auto Model, *accessed: 24.03.2022*

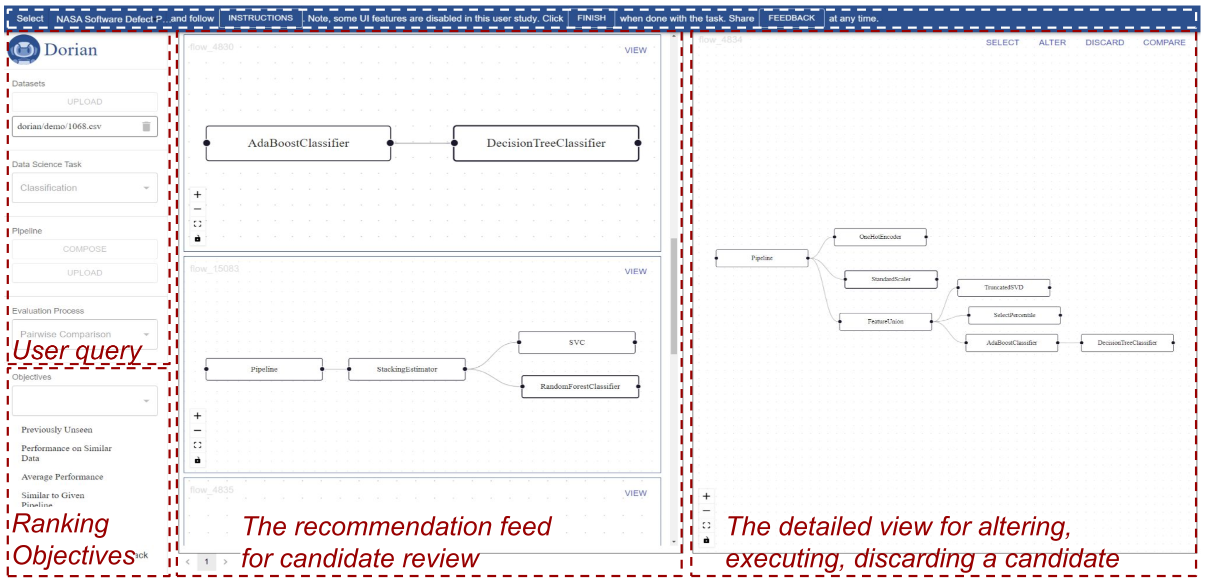

**Figure 3: DORIAN's Web-based User Interface**

❹ *Candidate Execution and Pairwise Comparison.* When *Alice* finds attractive pipeline candidates, they can choose to execute them and receive the results for candidate comparison. A pairwise comparison window (not shown in Figure 3) allows to compare the pipeline structure and its output side-by-side against the alternative candidate. The choice between two candidates affects the recommendations in the next recommendation round.

**Extensibility.** DORIAN is extensible to new operators, evaluation processes and even ranking objectives. To demonstrate this extensibility the audience will be able to do the following: (1) They can add new operators by using the code editor and annotating the DS task; (2) They can design DS pipelines that do not have quantifiable evaluation metrics by choosing "pairwise comparison" as the evaluation process - this allows them to view two pipelines and their corresponding results side-by-side and make a binary decision which candidate performs better; (3) They can update the number of objectives and their order to fine-tune the recommendation feed to their needs. For example, by prioritizing "previously unseen" over "better-performing" pipelines, the audience will be able to see new pipelines that DORIAN did not suggest before.

**User Experience.** We expect the audience to be able to easily navigate DORIAN's user-friendly interactive GUI and quickly construct their first DS pipelines. We have confirmed that with a preliminary user study we have conducted using a group of 6 PhD candidates with varying levels of expertise in data science. Based on the study, users perceive the tool as useful in scenarios where they want to explore new DS operators or pipeline candidates that might be relevant to their query and in scenarios where they want to receive a pool of potentially relevant pipeline candidates fast, without writing code. Furthermore, they appreciated the "drill-down" exploration opportunity and the chance to quickly get the overview of pipeline candidates that they did not necessarily know before. All end-users found pipelines that had competitive predictive performance to other baselines (see Figure 2) within few minutes.

## ACKNOWLEDGMENTS

This work was funded by the HEIBRiDS graduate school, with the support of the German Ministry for Education and Research as BIFOLD - Berlin Institute for the Foundations of Learning and Data (01IS18025A and 01IS18037A), the Software Campus Program (01IS17052), and Ahold Delhaize. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

## REFERENCES

[1] B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. 2018. Intelligent assistance for data pre-processing. *Computer Standards & Interfaces* 57 (2018), 101–109.
[2] Bernd Bischl et al. 2021. OpenML Benchmarking Suites. In *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
[3] I. Drori et al. 2018. AlphaD3M: Machine learning pipeline synthesis. In *AutoML Workshop at ICML*.
[4] M. Feurer et al. 2019. Auto-sklearn: efficient and robust automated machine learning. In *Automated Machine Learning*. Springer, Cham, 113–134.
[5] Y. Gil et al. 2018. P4ML: A phased performance-based pipeline planner for automated machine learning. In *ICML'18 AutoML Workshop*.
[6] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems* 212 (2021), 106622.
[7] Robert B. Miller. 1968. Response Time in Man-Computer Conversational Transactions. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I* (San Francisco, California) *(AFIPS '68 (Fall, part I))*. ACM, New York, NY, USA, 267–277. https://doi.org/10.1145/1476589.1476628
[8] P. Nguyen, M. Hilario, and A. Kalousis. 2014. Using Meta-Mining to Support Data Mining Workflow Planning and Optimization. *J. Artif. Int. Res.* 51, 1 (Sept. 2014), 605–644.
[9] R.S. Olson and J.H. Moore. 2016. TPOT: A tree-based pipeline optimization tool for automating machine learning. In *ICML'16 AutoML Workshop*. JMLR, 66–74.
[10] Z. Shang et al. 2019. Democratizing Data Science through Interactive Curation of ML Pipelines. In *SIGMOD'19* (Amsterdam, Netherlands). ACM, 1171–1188. https://doi.org/10.1145/3299869.3319863
[11] M.D. Wever, F. Mohr, and E. Hüllermeier. 2018. Ml-plan for unlimited-length machine learning pipelines. In *ICML'18 AutoML Workshop*.
[12] Q. Yao et al. 2018. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306* (2018).