SHORT-PAPER

DANTE: Hybrid AI System for Context-Aware Interpretable Feature Engineering

MOHAMED BOUADI, Laboratory of Informatics Paris Descartes, Paris, Ile-de-France, France

ARTA ALAVI, SAP SE, Walldorf, Baden-Wurttemberg, Germany

SALIMA BENBERNOU, Laboratory of Informatics Paris Descartes, Paris, Ile-de-France, France

MOURAD OUZIRI, Laboratory of Informatics Paris Descartes, Paris, Ile-de-France, France

Open Access Support provided by:

Laboratory of Informatics Paris Descartes

SAP SE

Published: 22 June 2025

Citation in BibTeX format

SIGMOD/PODS '25: International Conference on Management of Data
June 22 - 27, 2025
Berlin, Germany

Conference Sponsors:
SIGMOD

https://doi.org/10.1145/3722212.3725092
ISBN: 9798400715648

DANTE: Hybrid AI System for Context-Aware Interpretable Feature Engineering

Mohamed Bouadi
Université Paris Cité
LIPADE
Paris, France
mohamed.bouadi@u-paris.fr

Arta Alavi
SAP France
Levallois-Perret, France
arta.alavi@sap.com

Salima Benbernou
Université Paris Cité
LIPADE
Paris, France
salima.benbernou@u-paris.fr

Mourad Ouziri
Université Paris Cité
LIPADE
Paris, France
mourad.ouziri@u-paris.fr

Abstract

Machine learning has become essential in data management, enabling accuracy and efficiency in handling complex datasets. For structured data, however, feature engineering is required to enhance predictive performance but remains a time-consuming, domain-specific task. Moreover, as ML systems are increasingly adopted, ensuring interpretability, particularly for domain experts, has become important. To address these challenges, we proposed *KRAFT*, an automated feature engineering (AutoFE) approach that combines a neural generator for feature transformation with a knowledge-based discriminator to ensure interpretability through symbolic reasoning. We have also empirically shown that *KRAFT* outperforms SOTA approaches in both accuracy and interpretability. In this demonstration, we introduce *DANTE*, an interactive tool based on *KRAFT* to facilitate feature generation and model evaluation on structured data. Its no-code interface empowers users to generate meaningful, interpretable features for real-world applications. A demonstration video of DANTE is available at https://DANTE.com.

CCS Concepts

• **Computing methodologies** → **Feature selection**; **Reinforcement learning**; **Description logics**.

Keywords

Automated Feature Engineering, Deep Reinforcement Learning, Interpretable ML, Knowledge Graphs

ACM Reference Format:
Mohamed Bouadi, Arta Alavi, Salima Benbernou, and Mourad Ouziri. 2025. DANTE: Hybrid AI System for Context-Aware Interpretable Feature Engineering. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3722212.3725092

SIGMOD-Companion '25, Berlin, Germany

ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3725092

1 Introduction

In the past decade, ML has emerged as a transformative tool in data management, providing automated solutions that enhance scalability, accuracy, and efficiency in handling complex datasets. This success is attributed to the experience of data scientists who leverage domain knowledge to extract useful patterns. This crucial task is known as Feature Engineering (FE). However, defining and training an ML model using existing tools is still challenging.

Example 1 (Feature Engineering). *Consider the problem of predicting heart disease of patients based on their attributes (e.g. height, weight, etc). While these raw features provide valuable information, a more useful feature would be the Body Mass Index, BMI* $= \frac{weight}{height^2}$*, which can be derived using square and division.*

1.1 Challenges

The first challenge **C1** of FE pertains to the combinatorial search space. Initially, this space is given by the following equation:

$$\mathcal{F} = \bigcup_{i=1}^{p} \left\{ \left\{ \bigcup_{1 \le s_1 \le .. \le s_i \le p} \{(f_{s_1}, .., f_{s_i})\} \right\} \times T_i \right\}, \tag{1}$$

where p is the number of features and $T_i \subseteq \mathcal{T}$ is the set of i-ary transformations. The number of elements in this search space is $|\mathcal{F}| = \sum_{i=1}^{p} (A_i^p \times |T_i|)$, where A_i^p is the i-permutation of p features. As this space grows exponentially, exhaustive search is not feasible.

The second challenge, **C2**, lies in the need for extensive domain knowledge to generate meaningful features. Extracting useful patterns from data requires deep domain expertise, making FE a labor-intensive task. It is even more challenging with diverse data sources which is often the case in practice. The third challenge, **C3**, concerns feature interpretability. ML models are only as interpretable as their features [10]. Achieving high performance while ensuring that domain experts trust the model outputs depends on selecting interpretable features. Despite the growing interest in interpretable ML, existing approaches struggle to discover features that balance accuracy with interpretability.

1.2 Contributions

In this work, we introduce *DANTE*, a system that enables users to generate interpretanle features, evaluate ML models, and compare their performance on structured data. *DANTE* is built upon *KRAFT* [2], a novel AutoFE approach that tackles the aforementioned challenges. To address **C1**, we formalize FE as an optimization problem that maximizes model accuracy. Instead of an exhaustive search, we employ Deep Reinforcement Learning (DRL), specifically Deep Q-Network (DQN) [7], to iteratively transform raw features. The model prioritizes transformations with higher rewards, efficiently generating high-quality features while avoiding the exponential search space explosion. For **C2**, we train **KRAFT** on diverse datasets across multiple domains and integrate domain-specific knowledge graphs (KGs). This enables our model to embed domain semantics directly into the feature generation process, ensuring that the extracted features capture meaningful patterns relevant to the data context. Finally, to address **C3**, we define feature interpretability by considering both semantics and structural properties. A knowledge-based discriminator evaluates interpretability using domain knowledge, filtering out non-interpretable features to ensure that the generated features are not only predictive but also understandable and trustworthy for domain experts.

DANTE provides an end-to-end workflow where users, such as domain experts and data scientists, can upload data, automatically handle missing values and inconsistencies, generate meaningful features, and evaluate them using SOTA ML models, all through an intuitive dashboard. Our demonstration also includes an "under-the-hood" scenario, allowing attendees to manually perform FE and experience its inherent complexity.

1.3 Related Work

AutoFE has gained significant attention, resulting in several approaches. The first one generates a large number of features followed by pruning and selection [4, 8]. These techniques suffer from performance bottlenecks due to the large number of generated features. Another approach evaluates the usefulness of new features through training and evaluation. For instance, NFS [3] applies RNN-based controllers to generate transformations for every raw feature separately. DIFER [9] takes a different approach by using an encoder-predictor-decoder framework to improve features over time. *FETCH* [5], uses a pre-trained policy network within an RL framework to automate FE. However, these methods can be slow due to extensive model training and fail to address the problem of feature interpretability.

The paper is organized as follows: in section 2 we describe *KRAFT* [2]. In section 3, we present an overview of our framework *DANTE*. Section 4 describes our demonstration scenarios.

2 Proposed Approach: *KRAFT*

In this section, we propose a definition of feature interpretability and feature engineering and then we briefly describe *KRAFT* [2].

2.1 Problem Definition

Building a model to generate interpretable features requires a clear definition of interpretability. In ML, interpretability refers to the "ability to explain or to present in understandable terms to a human". While ML relies on good data, even simple and interpretable ones, like regression, become difficult to understand with non-interpretable features [10]. Existing research primarily focuses on model interpretability, often overlooking whether the features themselves are understandable. Our work shifts this focus to *feature interpretability*, ensuring that ML systems remain transparent and meaningful from the data level up

Definition 2.1 (Feature Interpretability). *It is the ability of domain experts to comprehend and connect the generated features with their domain knowledge. This implies mapping new features to relevant intensional and extensional knowledge within the domain of interest to gain deeper insights into the training data of the model.*

Consequently, interpretable features should be humanly-readable and refer to real-world entities that domain experts can understand and reason about them.

Definition 2.2 (Feature Engineering). *Given a predictive problem on a tabular dataset, $D(F, y)$, consisting of: (i) a set of p features $F = \{f_1, .., f_p\}$; (ii) an applicable ML model, L, and a corresponding cross-validation performance metric $\mathcal{P}$ (e.g. F1-score); (iii) an interpretability function, $I_{KG} : F \rightarrow \{0, 1\}$ that returns 1 if the feature is interpretable according to the KG and 0 otherwise. We define a FE pipeline $\mathcal{T} = \{t_1, ..., t_m\}$ as a sequence of m transformations applied to F (e.g. $+, -, \times, \div, \min, \max, log$). The set of transformed features from F using $\mathcal{T}$ is denoted as $\hat{F}_{\mathcal{T}}$.*

The goal of AutoFE is to find the optimal FE pipeline $\mathcal{T}$ that generates the set $\hat{F}_{\mathcal{T}}$ which maximizes the performance $\mathcal{P}(L(\hat{F}_{\mathcal{T}}, y))$ and remains, as far as possible, close to the concepts known by domain experts, as shown in Equation 2:

$$\begin{aligned} \mathcal{T}^* = \arg\max_{\mathcal{T}} \mathcal{P}(L(\hat{F}_{\mathcal{T}}, y)) \\ s.t. \\ \prod_{\hat{f}_i \in \hat{F}_{\mathcal{T}}} I_{KG}(\hat{f}_i) = 1, \ \forall \hat{f}_i \in \hat{F}_{\mathcal{T}} \end{aligned} \tag{2}$$

2.2 KRAFT in Nutshell

KRAFT [2] consists of two parts: a *Generator* and a *Discriminator*.

2.2.1 Generator. The goal of the generator is to learn how to generate interpretable features without exhaustively exploring the search space. We model FE as a Markov Decision Process (MDP) with:

(1) **State:** *KRAFT* introduces a knowledge-driven MDP, using a semantic vectorization to generate the state vector [2].
(2) **Actions:** correspond to the set of transformations $\mathcal{T}$.
(3) **Reward:** reflects the performance improvement in a k-fold cross-validation step: $r_i = \mathcal{P}(L(\hat{F}_i, y)) - \mathcal{P}(L(\hat{F}_{i-1}, y))$.
(4) **Policy Network:** The generator is modeled by a policy network, $\pi : S \rightarrow Q(A)$, where Q estimates the expected cumulative reward. It uses two fully connected neural networks: (i) The main one $Q(s, a; \theta_i)$ estimates the cumulative reward for each possible action; (ii) The target network, $\hat{Q}(s, a; \hat{\theta}_i)$, stabilizes training. The goal is to find the best sequence of m

transformations by optimizing the Q-learning loss function:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} \hat{Q}(s', a'; \hat{\theta}_i) - Q(s, a; \theta_i)\right)^2\right] \quad (3)$$

Basically, at each step of the training, and iteratively until convergence, *KRAFT* receives the state representing the input dataset and feed it to a neural network. This latter estimates an intermediate reward for each possible transformation and selects the one that maximizes the long term reward. The selected transformation is then used to generate new features.

2.2.2 Knowledge-based Discriminator. The discriminator is a reasoning algorithm that determines whether a generated feature is interpretable based on domain knowledge encoded in a KG. It treats each new feature $\hat{f} \in \hat{F}_{\mathcal{T}}$ as a concept and performs subsumption reasoning over the KG. If $\hat{f}$ is inconsistent with existing concepts (i.e., $KG \models \hat{f} \equiv \bot$), it is deemed non-interpretable and discarded [2]. For the purpose of this demonstration, we developed a KG[1] with two types of knowledge: (i) **Domain-agnostic knowledge**, covering units of measurement, quantities (e.g., Physics, Geometry), semantic rules about arithmetic functions, and relationships to guide the aggregation of numerical features based on categorical concepts; (ii) **Domain-specific knowledge**, incorporating concepts from various domains (e.g., Healthcare, Banking, E-commerce). This KG is also linked to external knowledge sources like DBpedia [1] and INSEE, making it extensible to new domains.

2.3 KRAFT Results

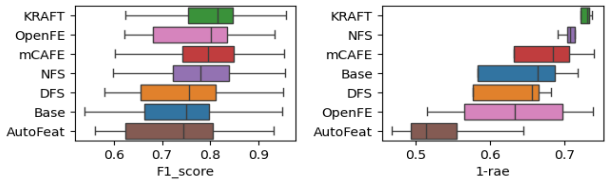


Figure 1: Comparison of KRAFT and 6 baseline methods.

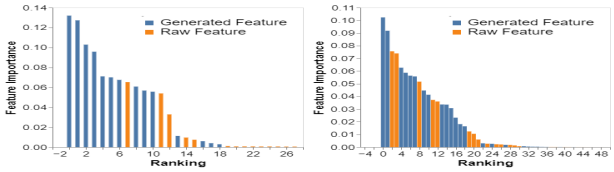
We compared *KRAFT* with SOTA AutoFE methods on multiple datasets, evaluating both accuracy (Figure 1) and interpretability (Figure 2). The results show that *KRAFT* consistently outperforms existing AutoFE approaches across all datasets. Additionally, *KRAFT*-generated features are systematically more important than raw features, enhancing model transparency and making its behavior more interpretable and trustworthy for users.

3 System Overview

DANTE is a stand-alone web application built with Python 3.9 and Streamlit. As shown in Figure 3, it consists of four main modules: (1) AutoFE, (2) Manual FE, (3) Evaluation, and (4) the GUI.

- **AutoFE module:** automates handling missing values, categorical features, and the generation of interpretable features using *KRAFT*.

[1]https://cutt.ly/bex8C2Cc



(a) NYC Taxi Ride dataset **(b) Home Credit Default Risk**

Figure 2: Feature importance of raw features (orange) versus KRAFT-generated features (blue).

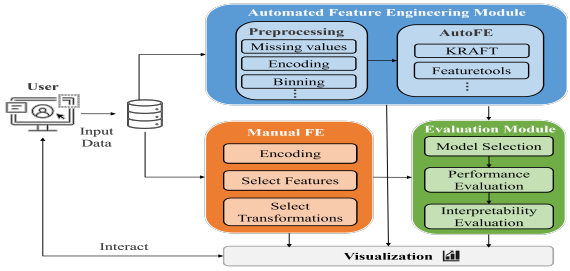


Figure 3: An overview of DANTE architecture.

- **Manual FE module:** allows users to manually apply transformations to generate new features interactively.
- **Evaluation module:** enables users to select SOTA ML models and cross-validation metrics, comparing *KRAFT* with other AutoFE methods.
- **GUI:** provides an intuitive interface where users can upload datasets, select ML models, performance metrics and choose FE methods via the sidebar shown in Figure 4.(A). Users can explore results via an interactive dashboard (Figure 4.C). They can manually apply transformations to generate features (Figure 4.B) or let the system generate features automatically. The GUI also presents feature evaluation metrics, including performance (Figure 4.D) and interpretability (Figure 4.E), using SHapley Additive exPlanations (SHAP) [6].

This design ensures an accessible, efficient, and transparent feature engineering workflow.

4 Demonstration Scenarios

The main objectives of this demonstration are to: (1) demonstrate the importance of FE in predictive modeling; (2) showcase the difficulties and time-consuming nature of manual FE; (3) emphasize the importance of feature interpretability; (4) demonstrate the efficiency of our approach *KRAFT* [2] in speeding up the workload of data scientists; and (5) engage participants in various scenarios to enable them to experience the challenges of FE process and appreciate the benefits of using automated tools.

[Scenario 1: Importance of FE and Effectiveness of *KRAFT*] This scenario demonstrates both the crucial role of FE in improving model performance and the effectiveness of *KRAFT* in generating high-quality features. Participants can either upload their own dataset or use the provided ones. They will first evaluate ML models with and without FE, highlighting its impact on accuracy. Then,

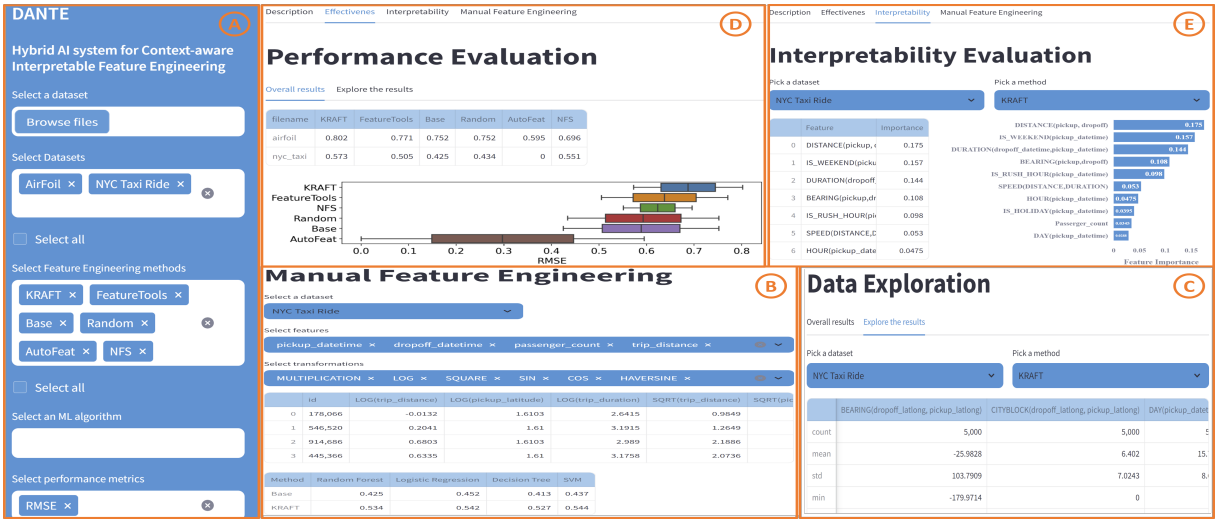


Figure 4: Interface of DANTE.

they will compare *KRAFT* with SOTA AutoFE methods, observing its superior performance across various datasets, models, and evaluation metrics. Unlike baseline methods, *KRAFT* maintains stable performance across different ML models. Participants can explore and analyze the generated datasets through interactive visualizations, gaining insights into the impact of FE on model accuracy.

[Scenario 2: Feature Usefulness and Interpretability] This scenario explores the quality of features generated by different approaches using **Information Gain** and **SHAP** [6]. SHAP values, being model-agnostic, measure each feature's contribution to predictions, while Information Gain quantifies how much a feature reduces uncertainty. Attendees will observe that *KRAFT* consistently produces features with higher SHAP values and greater Information Gain, demonstrating their strong impact on performance. In addition, attendees will notice that *KRAFT* produces human-readable and domain-relevant features that align with real-world entities. For example, when predicting "NYC taxi ride duration", meaningful features (e.g. *trip distance*, *is rush hours*, *weekday vs. weekend*) are generated. It is clear that the duration depends on the distance, since longer distances generally take more time. It also depends on the time of the day, rush hours usually have more traffic and thus longer trip durations. Consequently participants will see that *KRAFT* generates features that align with domain knowledge, ensuring that models remain interpretable and trustworthy.

[Scenario 3: Manual FE] In this final scenario, participants will experiment with manual feature engineering to improve predictive model accuracy. Given a dataset and a list of transformations (e.g., Logarithm, Addition, Division), they will intuitively select and apply transformations to different features, then evaluate their impact on ML models. Initially, they may apply transformations randomly and observe the results. As they refine their approach with more meaningful, domain-specific features, model accuracy will improve. For example, when predicting patient no-shows for medical appointments, generating features like *Duration* (time between scheduling and appointment), *Is Weekend*, and *Is Holiday* leads to a significant accuracy improvement compared to basic arithmetic transformations. A deeper analysis reveals that longer durations between appointment scheduling and the actual appointment correlate with more no-shows, and patients are less likely to miss appointments on weekends or holidays. Through this exercise, attendees will gain first-hand experience with the difficulties and challenges involved in identifying the most significant features that contribute to the model's performance and will gain an appreciation for automation tools.

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*. Springer, 722–735.

[2] Mohamed Bouadi, Arta Alavi, Salima Benbernou, and Mourad Ouziri. 2025. KRAFT: Leveraging Knowledge Graphs for Interpretable Feature Generation. In *International Conference on Web Information Systems Engineering*. Springer, 384–399.

[3] Xiangning Chen, Qingwei Lin, Chuan Luo, Xudong Li, Hongyu Zhang, Yong Xu, Yingnong Dang, Kaixin Sui, Xu Zhang, Bo Qiao, et al. 2019. Neural feature search: A neural architecture for automated feature engineering. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 71–80.

[4] James Max Kanter and Kalyan Veeramachaneni. 2015. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 1–10.

[5] Liyao Li, Haobo Wang, Liangyu Zha, Qingyi Huang, Sai Wu, Gang Chen, and Junbo Zhao. 2023. Learning a data-driven policy network for pre-training automated feature engineering. In *The Eleventh International Conference on Learning Representations*.

[6] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[8] Tianping Zhang, Zheyu Aqa Zhang, Zhiyuan Fan, Haoyan Luo, Fengyuan Liu, Qian Liu, Wei Cao, and Li Jian. 2023. OpenFE: automated feature generation with expert-level performance. In *International Conference on Machine Learning*. PMLR, 41880–41901.

[9] Guanghui Zhu, Zhuoer Xu, Chunfeng Yuan, and Yihua Huang. 2022. DIFER: differentiable automated feature engineering. In *International Conference on Automated Machine Learning*. PMLR, 17–1.

[10] Alexandra Zytek, Ignacio Arnaldo, Dongyu Liu, Laure Berti-Equille, and Kalyan Veeramachaneni. 2022. The need for interpretable features: motivation and taxonomy. *ACM SIGKDD Explorations Newsletter* 24, 1 (2022), 1–13.