Knowledge-Enriched Machine Learning for Tabular Data

Juyong Kim JUYONGK@CS.CMU.EDU
Machine Learning Dept., Carnegie Mellon University, Pittsburgh, USA

Chandler Squires CSQUIRES@ANDREW.CMU.EDU
Machine Learning Dept., Carnegie Mellon University, Pittsburgh, USA

Pradeep Ravikumar PRADEEPR@CS.CMU.EDU
Machine Learning Dept., Carnegie Mellon University, Pittsburgh, USA

Editors: G. Pappas, P. Ravikumar, S. A. Seshia

Abstract

In this paper, we introduce the general framework of *knowledge-enriched machine learning*, for encoding and leveraging problem-specific deterministic knowledge, such as column descriptions in the tabular setting. We focus on a paradigmatic use case: supervised learning problems on tabular data. As a first step in this direction, we introduce a simple yet flexible encoding of such deterministic information in the form of *concept kernels*, and describe meta-algorithms which leverage this particular encoding of prior knowledge. To ground future research, we introduce KE-TALENT, a novel benchmarking suite for kernel-enriched supervised learning on tabular data, adapted from the recently-introduced TALENT benchmark to include concept kernels and other metadata for each dataset. Finally, to demonstrate the benefits of concept kernels, we provide results for several kernel-enriched versions of existing algorithms, also intended as a baseline for future research. Code is publicly available.

Keywords: Knowledge-enriched machine learning, tabular machine learning, concept kernel

1. Introduction

In areas such as vision and language, deep learning has achieved remarkable success, benefiting from a combination of massive datasets and domain-appropriate inductive biases encoded into model architectures (Krizhevsky et al., 2012; Hochreiter, 1997). However, in areas such as tabular, relational, and scientific machine learning, deep models lag behind tree-based methods like XGBoost (Grinsztajn et al., 2022). This is often blamed on smaller dataset sizes in such settings. But as we argue in this paper, another critical reason is that the typical deep learning methods for tabular data do not leverage rich sources of deterministic domain knowledge, such as column names. Note that doing so is particularly challenging given the heterogeneity across datasets.

Overall, in many settings, rich sources of domain knowledge are indeed available, but what is lacking is a general approach for encoding such domain knowledge into algorithmically-usable forms. In this paper, we propose a general framework, which we call *knowledge-enriched machine learning*, to bridge this gap. This framework, specifically geared toward tabular data, provides high-level scaffolding for a new class of machine learning problems; as a paradigmatic example, we describe the problem of *kernel-enriched supervised learning* which extends the standard machine learning paradigm by considering an additional *concept kernel* as input to supervised learning algorithms.

To facilitate research in knowledge-enriched learning, we introduce KE-TALENT, a benchmark that extends the subset of the TALENT benchmark (Ye et al., 2024a) by incorporating structured metadata. Our benchmark consists of eleven datasets spanning diverse tasks and domains, with

each dataset including column description and concept kernels derived from sentence embeddings to encode semantic relationship between columns. The codebase also provides training pipelines, enabling researchers to systematically compare different knowledge-enriched learning approaches.

Given a concept kernel that provides a notion of geometry over the individual table columns or attributes, a critical question is then how to translate to a notion of geometry over entire inputs (or table rows). As one class of approaches, we provide an implicit notion of geometry over inputs by considering various forms of smoothers of the input that smooth it with respect to the concept kernel. As another class of approaches, we explicitly construct so-called value kernels over inputs that are explicitly specified in terms of the concept kernel. Given geometry over inputs, we can then extract features and train performant supervised learning models.

Our experiments show that kernel-enriched models demonstrate competitive performance and offer complementary feature representations. These findings highlight both the potential and challenges of integrating deterministic knowledge into tabular learning.

The remainder of this paper is structured as follows. Section 2 introduces the knowledge-enriched learning framework, formalizing the use of concept kernels. Section 3 describes KE-TALENT benchmark, detailing dataset construction and concept kernel generation. Section 4 presents our geometric approaches for knowledge-enriched learning. Section 5 provides experimental results, highlighting key findings and limitations. Finally, Section 6 concludes with a discussion of future research directions.

2. Knowledge-Enriched Learning

We begin by introducing our general framework of *knowledge-enriched machine learning*. Broadly speaking, this framework focuses on machine learning algorithms which take structured forms of *deterministic information* as input, in addition to the standard input of a dataset.

Consider a standard supervised learning problem with input space $\mathcal{X}$ and output space $\mathcal{Y}$. A *supervised learning algorithm* is a (possibly random) mapping $\mathcal{A}\colon \mathcal{D} \mapsto \widehat{m}_{\mathcal{D}}$, where $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$ is a dataset and $\widehat{m}_{\mathcal{D}}\colon \mathcal{X} \to \mathcal{Y}$ is a prediction function. Put simply, a supervised learning algorithm $\mathcal{A}$ takes a dataset as input and returns a prediction function as output.

We can easily extend this definition to the general notion of a *knowledge-enriched supervised learning algorithm*, which again is a (possibly random) mapping $\mathcal{A}\colon (\mathcal{D}, \mathcal{I}) \mapsto \widehat{m}_{\mathcal{D}}$, where $\mathcal{I}$ is some structured form of deterministic information (domain knowledge). To specialize this framework, we must specify the structural form of the deterministic information $\mathcal{I}$. In choosing the structure of $\mathcal{I}$, one must carefully balance the following goals:

- **Flexibility:** The structure should be flexible enough to encode many different forms of commonly-available knowledge, such as logical rules, natural language descriptions, and relations from knowledge graphs.
- **Informativity:** The structure should be complex enough to carry problem-specific information.
- **Pragmatics:** The structure of $\mathcal{I}$ provides a pragmatic layer of abstraction between potentially highly-unstructured domain knowledge on one hand, and the practical constraints of algorithm design on the other.

In this paper, we focus on a simple but flexible encoding of deterministic information in the form *concept kernels*, designed with tabular data problems in mind.

2.1. Concept Domains and Values

Let $\mathcal{C}$ be an arbitrary set of concepts, e.g. in a table taken from a dating website's database, the concepts may correspond to column names, with $\mathcal{C} = \{\mathsf{age}, \mathsf{city}, \mathsf{gender}, \mathsf{headshot}, \mathsf{biography} \ldots\}$.

Each concept $c \in \mathcal{C}$ is associated with a *concept domain*, denoted $\mathcal{V}_c$, which contains the possible values of that concept. In general, the concept domains may be both *rich* and *heterogenous*, e.g. we may have $\mathcal{V}_{\mathsf{headshot}} = [0,1]^{64\times64\times3}$ as the set of all 64×64 RGB images, and $\mathcal{V}_{\mathsf{biography}}$ as the set of all strings under 1,000 characters.

In these terms, a row in a table corresponds to assigning each concept to a value in its domain. When $\mathcal{C}$ and $(\mathcal{V}_c)_{c\in\mathcal{C}}$ are clear from context, we define $\mathcal{V} := \prod_{c\in\mathcal{C}} \mathcal{V}_c$ as the set of all possible assignments, termed the *value space*.[1] We use bold letters for assignments, and unbolded letters for values at specific concepts, e.g. $\mathbf{s} \in \mathcal{V}$ is an assignment and $s(c)$ is the value for concept c.

2.2. Concept Kernels

To relate concepts to a dataset $\mathcal{D}$, we require a correspondence between the concepts $\mathcal{C}$ and the input and output spaces $\mathcal{X}$ and $\mathcal{Y}$. Let $\mathcal{C}_{\text{in}} \subset \mathcal{C}$ be a set of *input concepts* and $\mathcal{C}_{\text{out}} \subset \mathcal{C}$ be a set of *output concepts*. Then, we assume that $\mathcal{X} = \prod_{c\in\mathcal{C}_{\text{in}}} \mathcal{V}_c$ and $\mathcal{Y} = \prod_{c\in\mathcal{C}_{\text{out}}} \mathcal{V}_c$, i.e., each $\mathbf{x} \in \mathcal{X}$ assigns each concept to a value in its domain, with $x(c) \in \mathcal{V}_c$ denoting the value assigned to concept c.

Finally, given a set of concepts $\mathcal{C}$, a *concept kernel* on $\mathcal{C}$ is a symmetric function $k\colon \mathcal{C} \times \mathcal{C} \to \mathbb{R}$. We are now ready to define a specific form of knowledge-enriched algorithm.

Definition 1 *A* kernel-enriched supervised learning algorithm *is a (possibly random) mapping $\mathcal{A}\colon \mathcal{D} \times k \mapsto \widehat{m}_{\mathcal{D}}$, where k is a concept kernel over $\mathcal{C}$, and each point in $\mathcal{D}$ belongs to the value space $\mathcal{V} = \prod_{c\in\mathcal{C}} \mathcal{V}_c$.*

It is crucial to note that k is a kernel over *concepts* rather than a kernel over *values*, i.e. in the tabular setting, k measures the similarity between columns, not the similarity between rows. This fact distinguishes our setup from the traditional usage of kernels in machine learning, e.g. in Gaussian process regression.

Kernel-enriched Stochastic Processes In general, a kernel-enriched supervised learning algorithm is defined without any need to specify probabilistic assumptions on the dataset $\mathcal{D}$. However, for theoretical purposes, we must often specify a data-generating model for $\mathcal{D}$.

Hence, we define a *kernel-enriched stochastic process* on $\mathcal{C}$ as a pair $(k, \mathbf{S})$, where $k\colon \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ is a concept kernel, and $\mathbf{S} = (S(c))_{c\in\mathcal{C}}$ is a stochastic process, with the random variable $S(c)$ taking values in the domain $\mathcal{V}_c$. Then, letting $\mathbf{X} = (S(c))_{c\in\mathcal{C}_{\text{in}}}$ and $\mathbf{Y} = (S(c))_{c\in\mathcal{C}_{\text{out}}}$, we may assume that each pair $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ in $\mathcal{D}$ is an independent sample from $\mathbb{P}(\mathbf{X}, \mathbf{Y})$.

Constructing and Using Concept Kernels Thus far, we have discussed the general framework of knowledge-enriched machine learning, and a more specific instantiation: kernel-enriched supervised learning. However, several details remain. First, *how can concept kernels be constructed from existing sources of problem-specific information?* Second, *how can algorithms leverage concept kernels to improve performance?*

Both questions are quite open-ended and constitute entire possible areas of research. To seed these areas, the remainder of the paper offers several basic starting points. First, in Section 3, we

1. In other contexts, such as physics and signal processing, the concept domains are homogeneous (i.e., $\mathcal{V}_c = \mathcal{V}_{c'}$ for all $c, c' \in \mathcal{C}$), in which case elements of $\mathcal{V}$ are called *configurations*, *states*, or *signals*.

Dataset Name	Domain	Task	# class	# sample	# num	# cat
Abalone	Biology	reg	-	4177	7(7)	1(1/3)
Diamonds	Geology	reg	-	53940	6(6)	3(3/20)
Parkinsons Telemonitoring	Healthcare	reg	-	5875	18(18)	1(1/2)
Student Performance	Education	reg	-	651	1(11)	29(19/53)
Communities and Crime	Social Science	reg	-	1994	102(102)	0(0)
Bank Customer Churn	Business	bincls	2	10000	6(6)	4(4/9)
German Credit Data	Business	bincls	2	1000	7(7)	13(13/54)
Taiwanese Bankruptcy	Business	bincls	2	6819	95(95)	0(0)
ASP-POTASSCO	Computer Science	multicls	11	1294	140(140)	1(1/2)
Internet Usage	Social Science	multicls	46	10108	1(1)	69(69/423)
Student Dropout	Education	multicls	3	4424	17(17)	17(17/218)

Table 1: **Statistics of the datasets in KE-TALENT**. The **Task** column denotes the task type (*reg* = regression, *bincls* = binary classification, *multicls* = multi-class classification). The **# sample** column denotes the number of samples (rows). The **# num** column denotes the number of numerical columns, and the **# cat** denotes the number of categorical columns. Numbers in parentheses indicate the number of columns and categories after preprocessing (See Appendix C).

introduce a benchmarking suite, which consists of several datasets along with potentially useful concept kernels. Then, in Section 4, we describe geometric approaches for leveraging concept kernels for the purpose of kernel-enriched machine learning.

3. KE-TALENT: Benchmark

While a few tabular machine learning benchmarks exist (Grinsztajn et al., 2022), none explicitly incorporate deterministic information about column semantics. To address this, we introduce KE-TALENT, a benchmark that extends a subset of the TALENT benchmark (Ye et al., 2024a), the most recent large-scale collection of tabular datasets. Our benchmark enhances TALENT by including column descriptions and embeddings, facilitating the use of prior knowledge in ML models.

We selected eleven datasets from TALENT where descriptive column names or metadata are available from the original data sources. They cover a diverse range of tasks and configurations of numerical and categorical input features. The dataset selection was determined *prior* to running our method to prevent post-hoc selection bias. Table 1 provides an overview of the dataset statistics.

To facilitate the use of deterministic information, KE-TALENT includes the following for each dataset: (1) original and preprocessed dataset, (2) metadata for each concept (column), (3) raw sentence embeddings of each concept (Reimers and Gurevych, 2019), (4) several pre-computed concept kernels, and (5) code for preprocessing, training, and evaluation. The benchmark will be continuously expanded to further enhance its scope and applicability.

Concept kernels provided in KE-TALENT For a chosen dataset in our benchmark, let $\mathcal{C}$ denote its concepts, which are in one-to-one correspondence with column indices. As indicated, KE-TALENT includes concept embeddings $(\boldsymbol{\lambda}_c)_{c\in\mathcal{C}}$; these embeddings can be used to construct concept kernels $k(c, c')$ in several ways. We provide multipe types of concept kernels, including inner product, distance-based, and group-centered inner product kernels. Please refer to Appendix B for the details.

4. Geometric Approaches to Kernel-Enriched Learning

A concept kernel specifies only a geometry over the *concepts* $\mathcal{C}$ (i.e., columns of a table), whereas in learning, we need to specify an inductive bias over the *values* $\mathcal{V}$ (i.e., rows of a table). Thus, a critical question in kernel-enriched learning is how to go from a geometry over coordinates to a geometry over values. Here, we describe several classes of approaches which accomplish this goal.

Notation For simplicity, we assume $\mathcal{C}$ is finite, with input concepts ordered as $\mathcal{C}_{\text{in}} = \{c_1, c_2, \ldots, c_D\}$ for some $D \in \mathbb{N}$. From here, we only use the concept kernel over the input concepts $k\colon \mathcal{C}_{\text{in}} \times \mathcal{C}_{\text{in}} \to \mathbb{R}$. Finally, we will frequently represent k by a symmetric matrix $\mathbf{K} \in \mathbb{R}^{D\times D}$, where $(\mathbf{K})_{ij} = k(c_i, c_j)$.

We assume *homogeneous* input concept domains, with values in some finite-dimensional vector space, i.e., $\mathcal{V}_c = \mathbb{R}^B$ for all $c \in \mathcal{C}_{\text{in}}$, which enables vector-space operations[2]. We can think of each dimension b as a *channel*. Then, we can identify each input value $\mathbf{x} \in \mathcal{X}$ with a matrix $\mathbf{M}_{\mathbf{x}} \in \mathbb{R}^{D\times B}$ whose i^{th} row equals $x(c_i)$. This makes $\mathcal{X}$ a vector space, with $\mathcal{X} \cong \mathbb{R}^{D\times B}$. We use $\mathbf{x}$ and $\mathbf{M}_{\mathbf{x}}$ interchangeably and write $\mathbf{x}_b$ for the b^{th} channel of $\mathbf{x}$ across all D concepts, i.e. $\mathbf{x}_b = (\mathbf{M}_{\mathbf{x}})_{:,b}$.

4.1. Smoothing Approaches

In the first class of approaches, we use the concept kernel to *implicitly* specify a geometry over values by specifying a transformation that takes each input value $\mathbf{x} \in \mathcal{X}$ to a smoother value $\widetilde{\mathbf{x}} \in \mathcal{X}$.

Smoothness via kernel convolution A simple way to smooth $\mathbf{x}$ with respect to k is via kernel convolution in value space, i.e., via the transformation $(k * -)\colon \mathcal{X} \to \mathcal{X}$ defined as

$$(k * \mathbf{x})(c) := \sum\nolimits_{c' \in \mathcal{C}_{\text{in}}} k(c, c') \cdot x(c'), \tag{1}$$

Letting $\widetilde{\mathbf{x}} = (k * \mathbf{x})$, we can represent in matrices as $\mathbf{M}_{\widetilde{\mathbf{x}}} = \mathbf{K}\mathbf{M}_{\mathbf{x}}$. To preserve scale, we can apply either the row-normalized $\overline{\mathbf{K}}_{\text{row}} = \mathbf{D}^{-1}\mathbf{K}$, or the symmetric-normalized $\overline{\mathbf{K}} := \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}$ kernels, where $\mathbf{D}$ is the diagonal degree matrix $\mathbf{D}_{ii} = \sum_j \mathbf{K}_{ij}$. Notably, the value $\widetilde{\mathbf{x}} = \mathbf{D}^{-1}\mathbf{K}\mathbf{x}$ solves the following kernel-weighted least variance objective at each input concept c:

$$(\overline{k}_{\text{row}} * \mathbf{x})(c) = \operatorname*{arg\,min}_{v \in \mathcal{V}_c} \sum\nolimits_{c' \in \mathcal{C}_{\text{in}}} k(c, c') \cdot \|x(c') - v\|_2^2 \tag{2}$$

Such a convolution can also be performed in the spectral domain. If k has a spectral decomposition, i.e., $k(c, c') = \sum_{m=1}^{D} \lambda_m \langle \psi_m(c), \psi_m(c') \rangle$ for orthonormal eigenfunctions $\boldsymbol{\psi}_m\colon \mathcal{C}_{\text{in}} \to \mathbb{R}$ and eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D$. We use m as our index to emphasize that each component is equivalent to a *mode* in signal processing, with higher-order eigenfunctions (i.e., those associated with smaller eigenvalues) corresponding to less smooth functions. If $\mathbf{x}_b = \sum_{m=1}^{D} \alpha_{bm}\boldsymbol{\psi}_m$, then

$$\widetilde{\mathbf{x}}_b = \mathbf{K}\mathbf{x}_b = \sum\nolimits_{m=1}^{d} \beta_{bm}\boldsymbol{\psi}_m, \quad \text{where } \beta_{bm} = \lambda_m \alpha_{bm}, \tag{3}$$

which is smoother since the higher-order eigenfunctions are attenuated, i.e., their associated eigenvalues are taken closer to zero. Thus, kernel convolution acts a soft low-pass filter.

2. $B = 1$ in typical tabular data, but we introduce B to generalize notation for cases where feature encodings are applied. If the input concept domains are heterogeneous, we can pre-process to make them homogeneous, as in Appendix C.

Smoothness via regularization Alternatively, one can use the concept kernel k to construct a smoothness penalty $R\colon \mathcal{X} \to \mathbb{R}$, and solve the regularized least squares problem

$$\widetilde{\mathbf{x}} = \arg\min_{\mathbf{v}\in\mathcal{X}} \sum\nolimits_{c\in\mathcal{C}_{\text{in}}} \|x(c) - v(c)\|_2^2 + \xi R(\mathbf{v}), \tag{4}$$

for a hyperparameter $\xi \geq 0$. For example, if k is a Mercer kernel, so that all eigenvalues in its spectral decomposition are nonnegative, then k induces the norm $R_{\text{norm}}(\mathbf{v}) := \|\mathbf{M}_{\mathbf{v}}^{\top}\mathbf{K}^{-1}\mathbf{M}_{\mathbf{v}}\|_2^2$. In this case, the optimum of Equation (4) is $\widetilde{\mathbf{x}}$ such that $\mathbf{M}_{\widetilde{\mathbf{x}}} = \mathbf{K}(\mathbf{K} + \xi I)^{-1}\mathbf{M}_{\mathbf{x}}$. In spectral terms, we can again view this procedure as a soft low-pass filter, since

$$\widetilde{\mathbf{x}}_b = \sum\nolimits_{m=1}^{D} \beta_{bm}\boldsymbol{\psi}_m, \quad \text{where } \beta_{bm} = \frac{\lambda_m}{\xi + \lambda_m}\alpha_{bm}. \tag{5}$$

Alternatively, we can use the kernel-weighted distance a smoothness penalty, i.e., $R_{\text{lap}}(\mathbf{v}) := \sum_{c,c'\in\mathcal{C}} \overline{k}(c,c') \cdot \|v(c) - v(c')\|_2^2$, where $\overline{k}(c,c') = d(c)^{-1/2}k(c,c')d(c')^{-1/2}$. In this case, we perform and analyze in the spectral domain of $\overline{\mathbf{K}}$. The optimum of Equation (4) is $\widetilde{\mathbf{x}}$ such that $\mathbf{M}_{\widetilde{\mathbf{x}}} = ((1+\xi)I - \xi\overline{\mathbf{K}})^{-1}\mathbf{M}_{\mathbf{x}}$, which has the spectral characterization

$$\widetilde{\mathbf{x}}_b = \sum\nolimits_{m=1}^{D} \beta_{bm}\boldsymbol{\psi}_m, \quad \text{where } \beta_{bm} = \frac{1}{1 + \xi - \xi\lambda_m}\alpha_{bm}, \tag{6}$$

where $\boldsymbol{\psi}_m$ and λ_m are from the eigen-decomposition of $\overline{\mathbf{K}}$.

General spectral transforms As noted, kernel convolution, norm regularization, and Laplacian regularization all amount to soft low-pass filtering with different filters; see Equations (3), (5) and (6), respectively. When the concept kernel k is a Mercer kernel, we can view all of these approaches as convolution with a modified version of k. In particular, for a function $s\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, we call

$$k_s(c,c') := \sum\nolimits_{m=1}^{D} s(\lambda_m)\langle\psi_m(c), \psi_m(c')\rangle \tag{7}$$

a *spectrally transformed* concept kernel. Holding to the interpretation that smaller eigenvalues are associated with less smooth eigenfunctions, we focus on functions s which attenuate smaller eigenvalues, which we call *attenuating spectral transforms*.[3] Equation (5) and (6) are such transforms.

4.2. Value Kernel Approaches

In the second class of approaches, we explicitly specify a geometry over the input space $\mathcal{X}$ by constructing a *value kernel* $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Given a concept kernel k, we start by using its spectral decomposition to construct a corresponding feature map $\boldsymbol{\Phi}\colon \mathcal{C}_{\text{in}} \to \mathbb{R}^D$:

$$\boldsymbol{\Phi}(c) := \big(\sqrt{\lambda_1}\cdot\psi_1(c), \sqrt{\lambda_2}\cdot\psi_2(c), \ldots, \sqrt{\lambda_D}\cdot\psi_D(c)\big), \tag{8}$$

so that $k(c,c') = \langle\boldsymbol{\Phi}(c), \boldsymbol{\Phi}(c')\rangle$. This feature map projects each concept $c \in \mathcal{C}_{\text{in}}$ into a D-dimensional eigenspace associated with the concept kernel: $\boldsymbol{\phi}_m\colon c \mapsto \sqrt{\lambda_m}\psi_m(c)$ is the m^{th} component of $\boldsymbol{\Phi}$.

Given such a feature map $\boldsymbol{\phi}_m$, an input value $\mathbf{x} \in \mathcal{X}$ can be represented in terms of these concept features $\mathbf{x}_b = \sum_{m=1}^{D}(\alpha_{bm}/\sqrt{\lambda_m})\boldsymbol{\phi}_m$. This decomposition by $\boldsymbol{\phi}_m$ induces a value kernel

$$K(\mathbf{x}_b, \mathbf{x}'_b) = \sum\nolimits_{m=1}^{D} \frac{\alpha_{bm}}{\sqrt{\lambda_m}}\frac{\alpha'_{bm}}{\sqrt{\lambda_m}} = \sum\nolimits_{m=1}^{D} \frac{1}{\lambda_m}\langle\mathbf{x}_b, \boldsymbol{\psi}_m\rangle\langle\mathbf{x}'_b, \boldsymbol{\psi}_m\rangle = \langle\varphi(\mathbf{x}_b), \varphi(\mathbf{x}'_b)\rangle, \tag{9}$$

3. Formally, such functions are monotonically increasing

where φ is the *value feature map*, and we use the inner product $\langle \mathbf{f}, \mathbf{g} \rangle = \sum_{c \in \mathcal{C}_{\text{in}}} f(c) \cdot g(c)$ for any two functions $\mathbf{f} \colon \mathcal{C}_{\text{in}} \to \mathbb{R}$ and $\mathbf{g} \colon \mathcal{C}_{\text{in}} \to \mathbb{R}$.

Different spectral transformations $s(\cdot)$ from the smoothing approach lead to different value kernels, but all share the same eigenfunctions $\boldsymbol{\psi}_m$. Specifically, the choice of spectral transformation $s(\cdot)$ only introduces a point-wise scaling to each value feature component:

$$\varphi_s(\mathbf{x}_b) = \left(\frac{1}{\sqrt{s(\lambda_1)}} \langle \mathbf{x}_b, \psi_1 \rangle, \frac{1}{\sqrt{s(\lambda_2)}} \langle \mathbf{x}_b, \psi_2 \rangle, \cdots, \frac{1}{\sqrt{s(\lambda_D)}} \langle \mathbf{x}_b, \psi_D \rangle \right). \tag{10}$$

In practice, we set $s(\cdot) = 1$ when transforming input into value feature. This particular choice corresponds exactly to the Fourier coefficients with respect to the concept kernel.

Combining value kernels As described, difference choices of concept kernel k lead to different value kernels K. We can can combine entire value kernels K_1 and K_2 with *value* feature maps $\boldsymbol{\Phi}_1 \colon \mathcal{X} \to \mathbb{R}^{D \times B}$ and $\boldsymbol{\Phi}_2 \colon \mathcal{X} \to \mathbb{R}^{D \times B}$. We can concatenate the value feature maps, resulting in the value kernel $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}')$. Alternatively, we can convolve the value kernels as $K(\mathbf{x}, \mathbf{x}') = (K_1 * K_2)(\mathbf{x}, \mathbf{x}') := \int_{\mathcal{X}} K_1(\mathbf{x}, \mathbf{v}) \cdot K_2(\mathbf{x}', \mathbf{v}) \cdot d\mathbf{v}$, which amounts to $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ interacting through integration over an intermediate space.

4.3. Partially Specified Concept Kernels

Thus far, we have united several potential approaches to kernel-enriched supervised learning, showing that we can use the provided concept kernel k to construct a value kernel K. However, in many cases, the concept kernel may be only partially specified, e.g. a binary sparsity pattern for k as $\mathbf{B} \in \{0, 1\}^{D \times D}$. Also, even if the concept kernel is fully available, we might wish to better model the concept relationship by binarizing the kernel and learning from data.

In this scenario of a binarized kernel, we present this kernel as edges in a graph, where each concept acts as a node. Graph Neural Networks (GNNs) (Kipf and Welling, 2017), specifically Graph Attention Networks (GATs) (Veličković et al., 2017; Brody et al., 2022), naturally accommodate this setting. At each GAT layer, attention weights can be dynamically learned based on both current node features and concept embeddings. We refer to this variant as *Concept Graph Attention Networks* (CGATs). Formally, a CGAT layer with node features $\mathbf{H}^{(l)} \in \mathbb{R}^{D \times B}$ and concept embeddings $(\boldsymbol{\lambda}_c)_{c \in \mathcal{C}}$ updates node features as:

$$\mathbf{h}_c^{(l+1)} = \sum_{c' : B_{cc'}=1} \alpha_{cc'} \mathbf{W}_t^{(l)} \mathbf{h}_{c'}^{(l)}, \text{ with } \begin{array}{l} \alpha_{cc'} = \exp(\text{score}_{cc'}^{(l)}) / \sum_{x : B_{cx}=1} \exp(\text{score}_{cx}^{(l)}) \quad \text{and} \\ \text{score}_{cc'}^{(l)} = \mathbf{a}^{(l)\top} \sigma(\mathbf{W}_s^{(l)} \mathbf{h}_c^{(l)} + \mathbf{W}_t^{(l)} \mathbf{h}_{c'}^{(l)}) + (\mathbf{W}_e^{(l)} \boldsymbol{\lambda}_c)^\top (\mathbf{W}_e^{(l)} \boldsymbol{\lambda}_{c'}), \end{array} \tag{11}$$

where σ is a nonlinear activation, such as LeakyReLU, and $\mathbf{W}_s^{(l)}$,$\mathbf{W}_t^{(l)}$,$\mathbf{W}_e^{(l)}$, and $\mathbf{a}^{(l)}$ are learnable parameters. This approach can adaptively emphasize significant concept pairs based on both learned representations and concept kernels. We can also apply a multi-head attention scheme to this.

4.4. Using Concept Kernels for Self-Supervised Learning

Concept kernels quantify similarities between the concepts or columns in tabular data. Leveraging this property, we construct a self-supervised learning (SSL) objective by defining a *value transition distribution*, which systematically swaps or replaces column values through a concept-based Markov

chain. Specifically, we row-normalize the concept kernel matrix[4] to form the concept transition matrix $\mathbf{T} = \mathbf{D}^{-1}\mathbf{K}$, which defines a Markov chain over concepts with a stationary distribution $\boldsymbol{\pi}_{\mathbf{K}}$. Given this, we define the value transition distribution as:

$$\mathbb{Q}(\mathbf{x}' \mid \mathbf{x}) = \sum_{c,c' \in \mathcal{C}} \boldsymbol{\pi}_{\mathbf{K},c} \mathbf{T}_{cc'} Q_{cc'}(x'_c | x_c), \tag{12}$$

where $Q_{cc'}$ is the value swap distribution to better capture complex or negative correlations.

Using this transition process, we generate multiple augmented views of training samples. SSL objectives, such as InfoNCE (Oord et al., 2018) or spectral contrastive loss (HaoChen et al., 2021), can then leverage these views to learn feature representations that inherently respect the geometric structure encoded by the concept kernel. Please refer to Appendix D for details of the SSL approach.

5. Experiments

In this section, we evaluate the approaches of kernel-enriched learning discussed in Section 4 on KETALENT, and compare them with strong tabular ML baselines. First, we describe the implemented models of knowledge-enriched supervised learning and the baselines. Then, we outline the evaluation methodology to ensure a fair comparison. Finally, we present and analyze the results.

5.1. Models

Smoothing models These models construct smoothed representations of input values by applying transformations derived from the concept kernel. Specifically, we explore three smoothing methods: (1) convolution using the row-normalized kernel, (2) smoothing via norm regularization, and (3) smoothing with a Laplacian penalty. After smoothing, the resulting representations are provided as input to an MLP prediction model[5].

Value kernel model This method projects an input row onto spectral components derived from the concept kernels. We use two concept kernels: the inner product kernel and the group-centered inner product kernel, where concept groups are identified via HDBSCAN clustering. The model architecture consists of projection matrices where the input is transformed into the spectral features, concatenation with the original input value, and an MLP model.

Partially specified concept kernel (CGAT) In this approach, we construct a graph of columns where edges are the top-p[6] highest absolute values from the concept kernel matrix $\mathbf{K}$. This graph is then utilized by Concept Graph Attention Network (CGAT) described in Section 4.3. The model architecture includes: a feature encoding layer from RealMLP to transform each column value into a dense embedding, multiple GCAT convolution layers implemented with PyTorch Geometric (Fey and Lenssen, 2019), a max-pooling operation over node features, and an MLP prediction head.

Self-supervised learning model Our SSL model consists of a feature encoder, which transforms an input row into a fixed-length feature vector, and an MLP prediction head. We choose FT-Transformer (Gorishniy et al., 2021) as the feature encoder. Training the SSL model is in two steps: initial

4. We assume $\mathbf{K}$ is non-negative, or pre-process it by clamping negative values to zero.
5. For the MLP architecture in smoothing and value kernel models, we re-implement RealMLP except data-driven initialization and dropout schedule.
6. p is selected by hyper-parameter search.

Dataset	**Abalone**	**Diamond**	**ParkTel**	**StuPerf**	**Crime**	**Churn**	**Credit**	**Taiwan**	**ASP**	**Internet**	**StuDrop**
Method \ Task	reg/RMSE(↓)					bincls/Acc(↑)			multicls/Acc(↑)		
RealMLP	2.1210	523.92	**0.7337**	2.9277	0.1381	0.8735	0.7157	0.9667	0.3861	0.5302	0.7655
CatBoost	2.1789	524.91	1.5994	2.9244	**0.1336**	0.8759	**0.7430**	0.9718	0.3815	**0.5358**	**0.7782**
TabR	2.1078	**513.53**	8.0521	2.9072	0.1437	0.8743	0.7240	0.9678	0.3750	0.5183	0.7493
FT-T	2.1078	532.83	8.3437	2.9642	0.1369	0.8709	0.7123	0.9674	0.3678	0.5348	0.7547
Smooth(kernel)	2.1718	938.03	2.4700	3.0651	0.1466	0.8657	0.7160	**0.9722**	0.3815	0.5042	0.7162
Smooth(norm)	2.0879	903.70	1.2112	2.9725	0.1401	0.8688	0.6960	0.9659	**0.4013**	0.5093	0.7579
Smooth(Laplacian)	2.0937	522.37	0.9530	**2.8926**	0.1397	**0.8765**	0.7193	0.9694	0.3900	0.5259	0.7673
Value kernel	**2.0825**	525.79	0.8676	2.9203	0.1394	0.8746	0.7157	0.9673	0.3761	0.5315	0.7665
CGAT	2.0876	677.33	1.4612	3.0397	0.1425	0.8764	0.7337	0.9675	0.3838	0.5275	0.7656
SSL	2.1584	534.12	1.1275	2.9178	0.1373	0.8757	0.7180	0.9655	0.3964	0.5327	0.7571

Table 2: **Performance on KE-TALENT benchmark** The table reports test set performance of various baselines and kernel-enriched learning models. Results are averaged over 15 runs after hyper-parameter tuning. **Bold** indicates the best-performing method per dataset. Underlined values denote methods statistically indistinguishable from the best method using Welch's t-test with $p = 0.05$ (↓: lower is better, ↑: higher is better).

contrastive learning on the augmented dataset and finetuning on the supervised task. See Appendix D for the implementation and training details.

Baselines To assess the effectiveness of our methods, we compare the best models in their respective model classes on the TALENT benchmark: **RealMLP** (Holzmüller et al., 2024), a multilayer perceptron tailored for tabular data with tuned hyper-parameters; **CatBoost** (Dorogush et al., 2018), a gradient boosting method handling categorical features via target statistics; **TabR** (Gorishniy et al., 2024), which integrates a k-nearest-neighbor-like component into deep learning models; and **FT-Transformer** (Gorishniy et al., 2021), combining feature tokenizers with Transformer layers.

5.2. Evaluation method

Our evaluation methodology directly follows TALENT to ensure a fair and consistent comparison across models. For each dataset and knowledge-enriched model, we perform hyper-parameter optimization using Optuna Akiba et al. (2019) for 100 trials based on validation performance. Each model is trained 15 times with the optimal hyper-parameters to report average test performance. For the SSL model, hyper-parameter search is conducted only during the fine-tuning step, optimizing the MLP prediction head and training parameters, while the contrastive learning step follows the default FT-Transformer configuration from the TALENT codebase. Appendix E details the tuning procedure and search space. For the baseline models, we use the results from the TALENT benchmark.

5.3. Results

Performance on KE-TALENT benchmark Table 2 presents performance of our kernel-enriched models and baselines on the KE-TALENT benchmark. CatBoost alone ranks best in four datasets, aligning prior findings that gradient boosting excels in tabular data settings. Nonetheless, kernel-enriched methods also demonstrate competitive performance, showing the potential of knowledge-

Dataset	CatBoost	SSL	CatBoost (+SSL Feature)
ASP-POTASSCO	0.3815	0.3964	**0.4067**

Table 3: **Combining SSL feature with CatBoost** Integrating SSL feature with CatBoost outperforms baseline CatBoost and SSL, suggesting that SSL representations complement tree-based models.

enriched learning: the smoothing models achieves top on four datasets, and the value kernel notably excels on Abalone dataset. Despite these successes, kernel-enriched models did not consistently surpass baselines, likely due to inefficiencies in concept kernel construction or how kernel information was integrated. Please refer to Appendix F for additional results and kernel visualizations.

Combining SSL features with CatBoost To assess the effectiveness of features learned from knowledge-enriched learning, we trained CatBoost using both the original tabular data and the feature from the pre-finetuning SSL encoder outputs on ASP-POTASSCO, where SSL ranked second. As shown in Table 3, this hybrid approach outperforms both baseline CatBoost and SSL, achieving the highest accuracy. This result suggests that SSL captures complementary metadata-driven features.

6. Discussion

Contributions In this paper, we outlined the general idea of *knowledge-enriched machine learning*, focusing on supervised learning algorithms that take an additional input in the form of a *concept kernel*. We proposed four meta-approaches that leverage concept kernels to inform their inductive biases. Additionally, we introduced KE-TALENT, a new benchmark for kernel-enriched supervised learning on tabular data, evaulated these approaches again strong tabular ML baselines. Our empirical results show that while kernel-enriched methods did not consistently outperform tree-based baseline, they demonstrated competitive performance and, in some cases, complementary feature representations.

Immediate future directions Our work opens the door to several future directions for knowledge-enriched machine learning in different domains. Here, we discuss a few directions of immediate importance, restricting our focus to the tabular data setting targeted by our benchmarking suite.

- **Improved kernel construction**: In our KE-TALENT benchmarking suite, we included several baseline kernels constructed from sentence embeddings. As the understanding of LLM embeddings improves, it may become possible to develop better methods for constructing kernels. Additionally, it may be interesting to use other forms of prior knowledge for constructing kernels, e.g. using knowledge graph embeddings (Ji et al., 2021).

- **Improved handling of heterogeneity**: Our datasets were carefully processed to ensure that the concept domains are relatively homogenous. In cases where this is not possible, or to further improve performance on these datasets, it may be necessary to enrich algorithms with other inputs, e.g. functions relating heterogeneous concept domains $\mathcal{V}_c$ and $\mathcal{V}_{c'}$.

- **Higher-order domain knowledge:** In some cases, binary relationships may not be sufficient to capture all domain knowledge. In these cases, it may be necessary to consider richer forms of knowledge enrichment and use those in methods which leverage higher-order structure, e.g. simplicial neural networks (Bodnar et al., 2021).

Acknowledgments

We acknowledge the support of AFRL and DARPA via FA8750-23-2-1015, ONR via N00014-23-1-2368, and NSF via IIS-1909816.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. SCARF: Self-supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations*, 2022.

Adrien Bardes, Jean Ponce, and Yann Lecun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022.

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. Weisfeiler and Lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, 2021.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. CatBoost: Gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 2021.

Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. TabR: Tabular deep learning meets nearest neighbors. In *International Conference on Learning Representations*, 2024.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 2022.

Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems*, 2021.

S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

David Holzmüller, Leo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned MLPs and boosted trees on tabular data. In *Neural Information Processing Systems*, 2024.

Roger A Horn and Charles R Johnson. *Matrix Analysis*. Cambridge University Press, 2012.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Daniel D. Johnson, Ayoub El Hanchi, and Chris J. Maddison. Contrastive learning can find an optimal basis for approximately view-invariant functions. In *International Conference on Learning Representations*, 2023.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, May 2023.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Kiho Park, Yo Joong Choe, Yibo Jiang, and Victor Veitch. The geometry of categorical and hierarchical concepts in large language models. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024a.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *International Conference on Machine Learning*, 2024b.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

B Schölkopf. Learning with kernels: support vector machines, regularization, optimization, and beyond, 2002.

Adly Templeton. *Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet*. Anthropic, 2024.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Zihao Wang, Lin Gui, Jeffrey Negrea, and Victor Veitch. Concept algebra for (score-based) text-controlled generative models. *Advances in Neural Information Processing Systems*, 2024.

Han-Jia Ye, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and De-Chuan Zhan. A closer look at deep learning on tabular data. *arXiv preprint arXiv:2407.00956*, 2024a.

Han-Jia Ye, Huai-Hong Yin, and De-Chuan Zhan. Modern neighborhood components analysis: A deep tabular baseline two decades later. *arXiv preprint arXiv:2407.03257*, 2024b.

Runtian Zhai, Bingbin Liu, Andrej Risteski, J Zico Kolter, and Pradeep Kumar Ravikumar. Understanding augmentation-based self-supervised representation learning via RKHS approximation and regression. In *International Conference on Learning Representations*, 2024.

Contents of Appendix

Symbol	Domain	Description
k	$\mathcal{C}_{\text{in}} \times \mathcal{C}_{\text{in}} \to \mathbb{R}$	Concept kernel (restricted to input concepts)
$\mathbf{x}$	$\mathcal{X}$	A function mapping each input concepts to a values in its domain
$x(c)$	$\mathcal{V}_c$	The function $\mathbf{x}$ evaluated at input concept c
D	$\mathbb{N}$	Number of input concepts
$\mathbf{K}$	$\mathbb{R}^{D\times D}$	Matrix representing the concept kernel (restricted to input concepts)
B	$\mathbb{N}$	Number of channels/bands in each concept domain
b	$\{1,\ldots,B\}$	An index for the b^{th} channel
$\mathbf{x}_b$	$\mathcal{C}_{\text{in}} \to \mathbb{R}$	The function mapping each concept to its value on the b^{th} channel
$\mathbf{M}_{\mathbf{x}}$	$\mathbb{R}^{D\times B}$	The matrix representing the function $\mathbf{x}$, with i^{th} row equal to $x(c)$
m	$\{1,\ldots,D\}$	An index for the m^{th} mode of an eigendecomposition
λ_m	$\mathbb{R}_{\geq 0}$	The m^{th} largest eigenvalue of k
$\boldsymbol{\psi}_m$	$\mathcal{C}_{\text{in}} \to \mathbb{R}$	The m^{th} eigenfunction of k
$\psi_m(c)$	$\mathbb{R}$	The m^{th} eigenfunction of k evaluated at input concept c
α_{bm}	$\mathbb{R}$	The m^{th} Fourier coefficient of the b^{th} channel of $\mathbf{x}$
β_{bm}	$\mathbb{R}$	The m^{th} Fourier coefficient of the b^{th} channel of $\widetilde{\mathbf{x}}$
ξ	$\mathbb{R}_{\geq 0}$	A regularization hyperparameter
$\boldsymbol{\Phi}$	$\mathcal{C}_{\text{in}} \to \mathbb{R}^D$	A concept feature map
$\boldsymbol{\phi}_m$	$\mathcal{C}_{\text{in}} \to \mathbb{R}$	The m^{th} component of the feature map $\boldsymbol{\Phi}$
φ	$\mathbf{x}_b \mapsto \mathbb{R}^D$	A value feature map
$\mathbf{W}^{(l)}_{\{s,t,e\}}$	$\mathbb{R}^{d_{\text{in}}\times d_{\text{out}}}$	Weight matrices in CGAT
$\mathbf{a}^{(l)}$	$\mathbb{R}^{d_{\text{out}}}$	Weight vector in CGAT

Table 4: **Notation.**

Appendix A. Notation

Table 4 summarizes the notation used throughout the paper by listing symbols, their mathematical domains, and concise descriptions for easy reference.

Appendix B. Constructing a Concept Kernel

As described in Section 4, concept kernels summarize deterministic information about columns and facilitate the incorporation of domain knowledge into tabular machine learning. The effectiveness of knowledge-enriched tabular machine learning heavily depends on the quality of these kernels, as they dictate the interaction between the columns. Here, we outline general approaches and key design choices for constructing concept kernels from column metadata.

B.1. From concept metadata to concept embeddings

From concept metadata, we extract concept embeddings $(\boldsymbol{\lambda}_c)_{c\in\mathcal{C}}$, where $\mathcal{C}$ is the set of concepts (or columns). Concept embeddings serves the foundation for constructing concept kernels. There are multiple ways to obtain these embeddings[7].

In a case there is a direct mapping from columns to the nodes in a knowledge graph, we can obtain node embeddings using a knowledge graph embedding method (Ji et al., 2021). However, such direct mappings are often unavailable. For example, in the "Communities and Crime" dataset of KE-TALENT, the columns represent measurements such as "percentage of population that is 12-29 in age", which do not correspond directly to a node in a knowledge graph. A more general approach is to extract sentence embeddings from a language model (Reimers and Gurevych, 2019), using textual descriptions of column names and metadata. This method provides a flexible way to obtain concept embeddings without requiring predefined mappings.

For the datasets in KE-TALENT, either column descriptions or descriptive column names are available in the original data sources. We extract sentence embeddings for each columns to obtain concept embeddings. Categorical columns are preprocessed into one-hot encoding, where each category requires a separate embedding. To generate meaningful presentations, we concatenate the column name with each category label. For example, the category single under the column marital status is converted into marital status is single. When a column description is available, it is appended to the column name before generating embeddings.

The choice of sentence embedding model influences the structure of the concept kernels. You can select a model that performs well across various sentence embedding tasks (Muennighoff et al., 2023), or one specialized for semantic textual similarity (STS) tasks. For sentence embedding models based on instruction tuning, the kernel matrix is often asymmetric. This is because similarity is computed using cosine similarity between query embeddings and document embeddings, with queries being prepended with instructions during embedding extraction. Another consideration is that in many sentence embedding models, similarity scores are more meaningful in terms of relative rankings rather than absolute values. We choose `all-mpnet-base-v2` model due to its widespread adoption and strong performance in general-purpose sentence embedding tasks, as noted in the Sentence-Transformer (Reimers and Gurevych, 2019) documentation.

We note that while our approach is general, there is ample room for improvement by refining column descriptions and selecting a more suitable sentence embedding model through analysis of the concept kernel matrix.

B.2. From concept embeddings to concept kernel

Given concept embeddings $(\boldsymbol{\lambda}_c)_{c\in\mathcal{C}}$, we construct concept kernels $k(c, c')$ to provide structured similarity between concepts (or columns). For each dataset, we provide the following concept kernels as baselines:

1. **Inner product:** Here, $k(c, c') = \langle \boldsymbol{\lambda}_c, \boldsymbol{\lambda}_{c'} \rangle$.
2. **Centered inner product:** Let $\overline{\boldsymbol{\lambda}} = \frac{1}{|\mathcal{C}|} \sum_{c\in\mathcal{C}} \boldsymbol{\lambda}_c$. Then $k(c, c') = \langle \boldsymbol{\lambda}_c - \overline{\boldsymbol{\lambda}}, \boldsymbol{\lambda}_{c'} - \overline{\boldsymbol{\lambda}} \rangle$.
3. **Exponential squared distance:** Here, $k(c, c') = \exp\left(-\|\boldsymbol{\lambda}_c - \boldsymbol{\lambda}_{c'}\|_2^2\right)$.

7. Concept kernels can be directly obtained without explicit concept embeddings, in rare cases where columns have a direct mapping to concepts in a knowledge graph with predefined similarities.

Each of these standard kernels corresponds to a different interpretation of the geometry of the embedding space (Schölkopf, 2002). Understanding the geometry of sentence embeddings is an active area of research (Park et al., 2024b,a; Templeton, 2024), and such research may pave the way to better ways of constructing kernels for our datasets.

While kernel choice is not the primary focus of our work, we do remark on an important phenomenon. Due to the nature of sentence embeddings, many of these constructions yield kernels which assign high similarity to concept pairs such as $c =$ the-color-is-red and $c' =$ the-color-is-blue. The associated random variables $X(c)$ and $X(c')$ will typically be highly dependent (in terms of mutual information), but they may take on very dissimilar values, e.g., they will tend to have a large *negative* correlation.

Depending on how a method uses the concept kernel, it may be helpful to adapt the kernel to encode such negative relationships. To this end, we provide for each dataset a *concept partition*, i.e. a division of $\mathcal{C}$ into disjoint subsets of concepts $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_E$. From such a partition, we construct an additional kernel:

4. **Group-centered inner product:** Let $\overline{\boldsymbol{\lambda}}_e = \frac{1}{|\mathcal{C}_e|} \sum_{c \in \mathcal{C}_e} \boldsymbol{\lambda}_c$. Then

$$k(c, c') = \begin{cases} \langle \boldsymbol{\lambda}_c - \overline{\boldsymbol{\lambda}}_k, \boldsymbol{\lambda}_{c'} - \overline{\boldsymbol{\lambda}}_k \rangle & c, c' \in \mathcal{C}_k \\ \langle \boldsymbol{\lambda}_c, \boldsymbol{\lambda}_{c'} \rangle & \text{otherwise} \end{cases}$$

The provided concept partitions can also be used for more complex kernel constructions, e.g. using subspace projections inspired by concept algebras (Wang et al., 2024). Note that, unlike kernels 1-3, this kernel is not necessarily positive definite; possibly limiting what algorithms are applicable.

Appendix C. Tabular Data Processing

In general, we pre-process tabular data to ensure robust training with ML models. Tabular datasets typically contain two types of columns: numerical (continuous or ordinal values) and categorical (discrete labels). Each type requires appropriate preprocessing to facilitate learning in a deep frameworks, and preprocessing is dependent on the architecture.

Numerical columns can have varying ranges, units and scales (*e.g.*, linear vs. logarithmic). To bring them to a comparable scale while keeping preprocessing simple, we apply column-wise standardization, *i.e.*, subtracting the mean and dividing by the standard deviation. Standardizing numerical features aligns well with extracting spectral components corresponding to smooth graph signals, or eigen functions, and stabilizes training. One can also choose other forms of normalization such as min-max scaling, or, in case the distribution is highly skewed, quantile transformation.

Categorical columns, being nominal, require separate handling. Simply assigning ordinal values to the category labels may introduce arbitrary order to the labels, potentially degrading performance when combined with models that assumes a meaningful order. To avoid this while applying to our various approaches (smoothing, value kernel, and CGAT model), we apply one-hot encoding, representing each category as an independent binary vector. During preprocessing, some of the categorical columns are moved into numerical columns when they represent ordinal values.

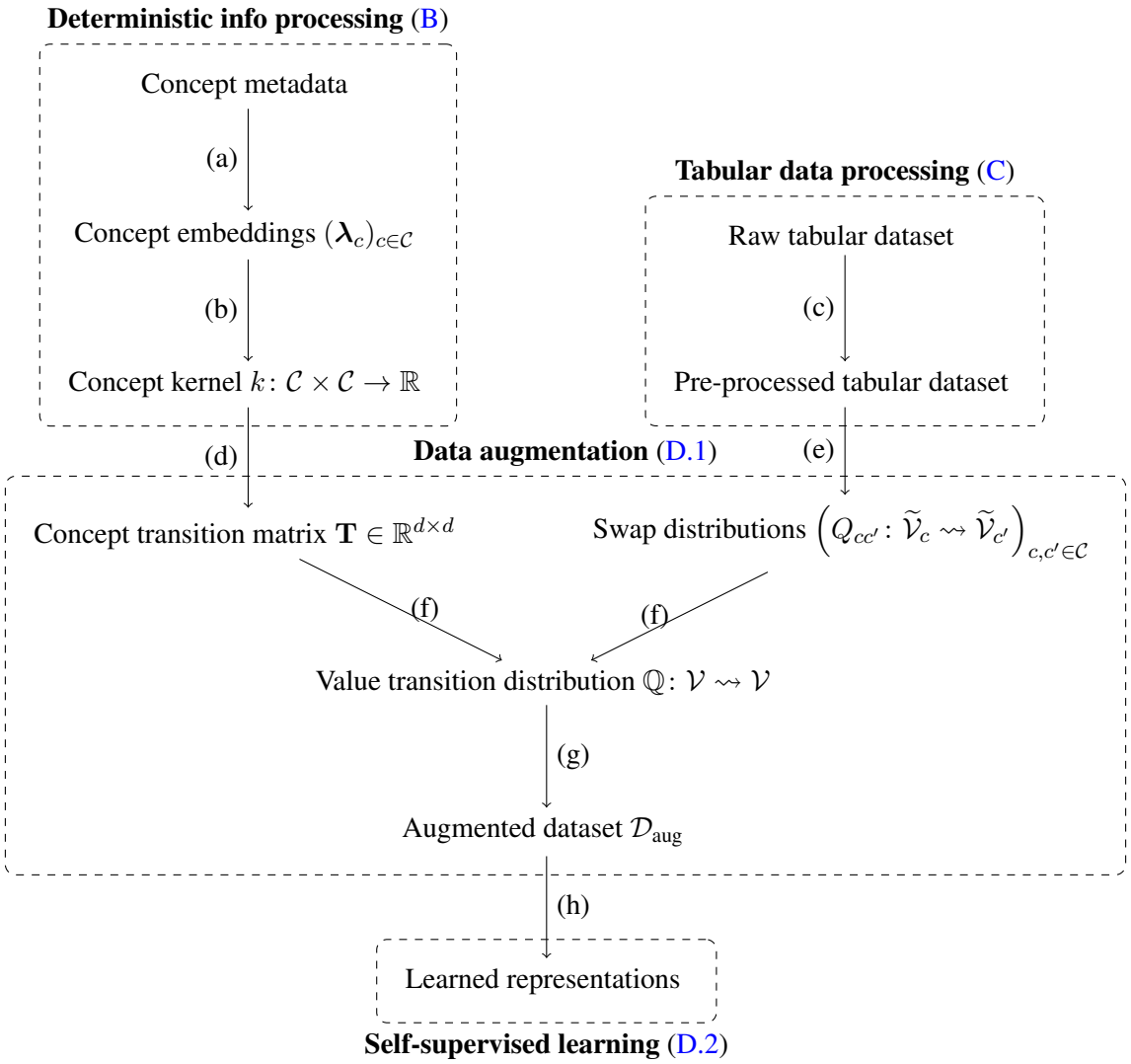

Figure 1: **Full pipeline of self-supervised learning** This figure illustrates the complete pipeline for knowledge-enriched supervised learning. The pipeline consists of multiple stages, including deterministic information processing, tabular data processing, data augmentation, and self-supervised learning. Please refer to the Sections B-D.2 for the detailed descriptions of each stage.

Appendix D. Self-supervised Learning

In this section, we describe the method, implementation details of the pipeline, and key design choices involved in building knowledge-enriched self-supervised learning models for KE-TALENT. The overarching objective of knowledge-enriched SSL model is to learn feature representations from both the raw tabular dataset and the associated concept metadata while leveraging the prior knowledge encoded in concept metadata to improve downstream task performance. Figure 1 illustrates the full pipeline and Algorithm 1 a step-by-step overview of the SSL training procedure.

Algorithm 1 ConceptSSL

Require: Concept kernel $k\colon \mathcal{C} \times \mathcal{C} \to \mathbb{R}$, dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$, losses $(\ell_{\text{SSL}}, \ell_{\text{task}})$, $M \in \mathbb{N}$
Ensure: Encoder $h : \mathcal{X} \to \mathbb{R}^d$, prediction head $g : \mathbb{R}^d \to \mathcal{Y}$

Value transition distribution
Compute the concept transition matrix $\mathbf{T}$ and the stationary distribution $\boldsymbol{\pi}_{\mathbf{K}}$ from k
Compute the value swap distribution $Q_{cc'}$ for each $c_i, c_j \in \mathcal{C}$.
Define the value transition distribution $\mathbb{Q}(\mathbf{x}'|\mathbf{x})$ using Equation 12
Dataset augmentation
Let $\mathcal{D}_{\text{aug}} = \varnothing$
for $m = 1, \ldots, M$ **do**
    Sample $\mathbf{x}$ and $\mathbf{x}'$ uniformly and independently from $\mathcal{D}$
    Generate augmented views $\mathbf{x}^1, \mathbf{x}^2 \sim \mathbb{Q}(\cdot|\mathbf{x})$ and a negative sample $\mathbf{x}^- \sim \mathbb{Q}(\cdot|\mathbf{x}')$.
    Add $(\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^-)$ to $\mathcal{D}_{\text{aug}}$
end for
Self-supervised learning
Train h on $\mathcal{L}_{\text{SSL}}(h) = \mathbb{E}_{(\mathbf{x}^1,\mathbf{x}^2,\mathbf{x}^-)\sim\mathcal{D}_{\text{aug}}}\big[\ell_{\text{SSL}}(f_\theta(\mathbf{x}^1), f_\theta(\mathbf{x}^2), f_\theta(\mathbf{x}^-))\big]$
Fine-tuning
Train h and g on $\mathcal{L}_{\text{task}}(h, g) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\big[\ell_{\text{task}}(g(h(\mathbf{x})), \mathbf{y})\big]$

First, we show how the concept kernel $k\colon \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ can be used to construct a value kernel $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ using a self-supervised approach. In the following subsections, we will follow the steps in the figure that are specific to the SSL approach, describing each steps in detail.

Constructing a value transition distribution To begin, assume that $\mathbf{K}_{\text{in}}$ has only nonnegative entries, let $\mathbf{D}$ be a diagonal matrix with $(\mathbf{D})_{ii} = \sum_{j=1}^{d_{\text{in}}} k_{ij}$, and let $\mathbf{T} := \mathbf{D}^{-1}\mathbf{K}_{\text{in}}$. Then $\mathbf{T}$ is a right stochastic matrix, i.e., its rows sum to one. We call $\mathbf{T}$ the *concept transition matrix*, since it defines a *concept Markov chain*, i.e., a Markov chain over concepts. Assuming that $\mathbf{T}$ is irreducible and aperiodic, this Markov chain has a unique stationary distribution which we can represent as a row vector $\boldsymbol{\pi}_{\mathbf{K}}$. With some abuse of notation, we will use $\boldsymbol{\pi}_{\mathbf{K}}$ to denote the stationary distribution, and we will use $\mathbf{T}_i$ to denote the transition distribution from concept c_i.

Now, we can use the concept Markov chain to define a Markov chain over the (potentially uncountable) set of input values $\mathcal{X}$. In particular, given any value $\mathbf{x} \in \mathcal{X}$, consider performing one or more iterations of the following steps:

$$\text{1. Sample } c_i \sim \boldsymbol{\pi}_{\mathbf{K}}. \qquad \text{2. Sample } c_j \sim \mathbf{T}_i. \qquad \text{3. Let } \mathbf{x}^1 \leftarrow \mathsf{swap}(\mathbf{x}\,;\, c_i, c_j).$$

Here, $\mathsf{swap}(\cdot\,;\, c_i, c_j)$ denotes the function which takes $\mathbf{x}$ as input and returns $\mathbf{x}'$ such that $x'(c) = x(c)$ if $c \notin \{c_i, c_j\}$, $x'(c_i) = x(c_j)$, and $x'(c_j) = x(c_i)$. Additionally, to handle (potentially negative) correlation between column values, we further transform the swapped values in $\mathbf{x}^1$ via *value swap distribution* $Q_{cc'}$, a (probabilistic) mapping from $\mathcal{V}_c$ to $\mathcal{V}_{c'}$. This process induces a transition distribution over values, which we call the *value transition distribution*, denoted $\mathbb{Q}(\mathbf{X}^1 \mid \mathbf{X})$:

$$\mathbb{Q}(\mathbf{x}' \mid \mathbf{x}) = \sum_{c,c' \in \mathcal{C}} \boldsymbol{\pi}_{\mathbf{K},c} \mathbf{T}_{cc'} Q_{cc'}(x'_c|x_c). \tag{13}$$

A related work is Bahri et al. (2022), which augments tabular columns by sampling from marginal distributions, whereas ours leverages concept kernels to incorporate column relationships.

Given the value transition distribution, one can perform augmentation-based self-supervised learning in a number of ways, e.g. for any anchor value $\mathbf{x} \in \mathcal{X}$ taken from our dataset, we can generate a positive sample $\mathbf{x}^1 \sim \mathbb{Q}(\cdot \mid \mathbf{x})$. Then, we can minimize contrastive training objective over these pairs to pre-train a value embedding function $h\colon \mathcal{X} \to \mathcal{Z}$ and use this representation in downstream algorithms. Intriguingly, this process can be seen as transforming each $\mathbf{x}$ by a generalized Fourier expansion in the basis associated with a specific kernel, as we now describe.

The value kernel and eigenspace extraction Given the value transition distribution $\mathbb{Q}(\mathbf{X}^1 \mid \mathbf{X})$ and a data distribution $\mathbb{P}(\mathbf{X})$, we define the value kernel $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as

$$K(\mathbf{x}^1, \mathbf{x}^2) := \frac{\int \mathbb{Q}(\mathbf{x}^1 \mid \mathbf{x}) \cdot \mathbb{Q}(\mathbf{x}^2 \mid \mathbf{x}) \cdot d\mathbb{P}(\mathbf{x})}{\mathbb{P}(\mathbf{x}^1) \cdot \mathbb{P}(\mathbf{x}^2)}, \tag{14}$$

also known as a *positive-pair kernel* (Johnson et al., 2023; Zhai et al., 2024). Hence, we see that a kernel k over concepts can be used to construct a kernel K over values, which more directly expresses an inductive bias over prediction functions $m\colon \mathcal{X} \to \mathcal{Y}$.

Fortunately, to utilize the value kernel K, we do not need to evaluate it, which would require estimating the data distribution $\mathbb{P}(\mathbf{X})$ and approximating the potentially intractable integral in the numerator. Instead, we may work directly with the eigenspace of this kernel, and there are several potential approaches to efficiently extracting the eigenspace, utilizing its density ratio nature.

A popular class of such approaches is based on *contrastive learning*. For example, HaoChen et al. (2021) define the *spectral contrastive loss*

$$\mathcal{L}_{\text{sc}}(h) := -2 \cdot \mathbb{E}\left[h(\mathbf{X}^1)^\intercal h(\mathbf{X}^2)\right] + \mathbb{E}\left[\left(h(\mathbf{X}^1)^\intercal h(\mathbf{X}^-)\right)^2\right], \tag{15}$$

where the joint distribution over $\left(\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^-\right)$ is given by

$$\int \mathbb{Q}(\mathbf{X}^1 \mid \mathbf{x}) \cdot \mathbb{Q}(\mathbf{X}^2 \mid \mathbf{x}) \cdot \mathbb{Q}(\mathbf{X}^- \mid \mathbf{x}') \cdot d\mathbb{P}(\mathbf{x}) \cdot d\mathbb{P}(\mathbf{x}').$$

They show that, when h ranges over all possible functions, the minimum of Equation (15) yields the true eigenspace of the positive-pair kernel.[8] In practice, h is parameterized by a neural network, and the empirical counterpart of Equation (15) is minimized, thus the eigenspace is not recovered perfectly. See Appendix D.2 for other options for self-supervised learning objectives.

D.1. Data augmentation

For steps **(a)**-**(c)** of Figure 1, we follow the procedures described in Appendix B and C. In the remainder of this section, we describe the data augmentation and feature representation learning steps specific to the SSL approach.

D.1.1. FROM CONCEPT KERNEL TO CONCEPT TRANSITION MATRIX

In **(d)**, we transform the concept kernel into the *concept Markov chain* that is needed in the value transition distribution. As described in Section 4.4, we row-normalize the concept kernel matrix $\mathbf{K}_{\text{in}}$ to obtain the transition matrix $\mathbf{T}$, from which the stationary distribution $\boldsymbol{\pi}_{\mathbf{K}}$ is derived. The

8. In particular, this follows from their Lemma 3.2, which relates the spectral contrastive loss to a low-rank approximation of the kernel. See also Table 1 of Johnson et al. (2023).

stationary distribution $\boldsymbol{\pi}_{\mathbf{K}}$ specifies the probability of selecting the concept to be swapped, while $\mathbf{T}$ determines the probability to sample to a new concept based on the current one.

To ensure the concept kernels are nonnegative, we clamp negative entries in $\mathbf{K}_{\text{in}}$ to zero. Alternatively, an element-wise transformation, such as exponentiation, can be applied to make all values positive while preserving their relative order. Note that the (cosine) similarity of sentence embeddings, which is the choice of concept kernel in our implementation, is better interpreted by relative order rather than by absolute values.

When selecting concepts to swap using the transition matrix $\mathbf{T}$, we handle numerical and categorical columns differently. When selected concept corresponds to a numerical column, we restrict to choose another numerical columns since we swap the values. In this case, a single transition affects two numerical columns.

For categorical columns, the selected concept represents one of the categories within that column (or the active category of one-hot encoded columns). There are several options for determining the replacement:

- **Swap categories between two categorical columns**: Choose another categorical column and exchange their active categories.
- **Replace the category within the same column**: Select a different category from the same column and replace the current category.
- **Replace the category with one from any categorical column**: Choose a category from any categorical column and substitute it for the current one.

Note the first option modifies two categorical columns, while the latter two affect only a single column.

D.1.2. LEARNING SWAP DISTRIBUTIONS

In **(e)**, we define how the actual swapping of column values occurs after columns are selected by the concept transition matrix. If we simply interchange the values between two columns, the process becomes equivalent to directly using the concept kernels as the correlation of column values. However, there are potentially more sophisticated methods that we can apply to better model the swap distribution $Q_{cc'} : \widetilde{\mathcal{V}}_c \rightsquigarrow \widetilde{\mathcal{V}}_{c'}$ as a probabilistic mapping.

For numerical columns, one straightforward approach to model $Q_{cc'}$ is to simply apply a linear transformation to the swapped column values. The weight and bias of this transformation can be pre-determined or trained through self-supervised learning. Possible strategies include:

- **Linear regression**: Fix the weights and bias to the solution of the linear regression. Under standardization, the weight corresponds to the correlation $\rho_{cc'}$ and the bias is set to zero.
- **Correlation sign**: Fix the weight to be the sign of the correlation and the bias to zero. This addresses the problem of negative value correlation between two close concepts.
- **Trainable weights**: Initialize the weight and bias to those from linear regression and allow them to be updated during training.

We chose the first option as the swap distribution on numerical columns. However, one can model $Q_{cc'}$ in a more complex manner, such as incorporating trainable weights or making it probabilistic by predictive posterior variance.

Since we adopt FT-Transformer Gorishniy et al. (2021) as the feature encoder, each category in categorical columns is represented as an embedding vector. Transition of categorical columns corresponds to changing one embeddings to another. For simplicity and to keep the number of trainable parameters small, we use the changed embedding directly without further transformations.

D.1.3. CONSTRUCTING THE VALUE TRANSITION DISTRIBUTION

In **(f)**, we construct the value transition distribution $\mathbb{Q}\colon \mathcal{V} \rightsquigarrow \mathcal{V}$ by integrating the concept transition distribution $\mathbf{T}$ with the value swap distribution $Q_{cc'}$. Specifically, given a concept c sampled from the stationary distribution $\pi_{\mathbf{K}}$ and its paired transition concept $c' \sim \mathbf{T}_c$, the transition of values follows $Q_{cc'}(x'_c|x_c)$. Formally, the value transition distribution is defined as

$$\mathbb{Q}(\mathbf{x}'|\mathbf{x}) = \sum_{c,c' \in \mathcal{C}} \pi_{\mathbf{K},c} \mathbf{T}_{cc'} Q_{cc'}(x'_c|x_c).$$

D.1.4. DATA AUGMENTATION

In **(g)**, we construct the augmented dataset $\mathcal{D}_{\text{aug}}$, we apply the value transition distribution $\mathbb{Q}$ to generate augmented views of samples in $\mathcal{D}$. By applying $\mathbb{Q}$ to a given input instance (or row) $\mathbf{x}$, we obtain a transformed sample $\mathbf{x}'$, which preserves the semantics while introducing controlled perturbations. In Section D.1, $\mathcal{D}_{\text{aug}}$ formulates a set of triplets $(\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^-)$, where $\mathbf{x}^1$ and $\mathbf{x}^2$ are positive pairs and $\mathbf{x}^-$ is a negative sample, as structure particularly useful for spectral contrastive loss and other triplet-based objectives. However, for InfoNCE loss where negative samples are implicitly drawn from the batch rather than explicitly drawn from the mini batch, so different formulation of $\mathcal{D}_{\text{aug}}$ is required.

For datasets with many columns, we allow multiple value transitions per sample, where the number of transitions is selected based on the number of columns. Appendix E describes the heuristics in determining the number of transitions per different dataset to ensure sufficient augmentation. Also, note that additional stochasticity can be incorporated into $\mathcal{D}_{\text{aug}}$ to enhance diversity, such as introducing noise into numerical columns.

D.2. Self-supervised representation learning

So far, we have obtained the augmented dataset $\mathcal{D}_{\text{aug}}$ from value transition distribution. In **(h)**, using this dataset, we perform contrastive representation learning to learn feature representation of instances and to perform supervised learning. The training process consist of two main steps:

- **Contrastive learning to learn features**: The feature encoder $h(\mathbf{x})$ is initially trained on $\mathcal{D}_{\text{aug}}$ using a contrastive learning objective to extract meaningful representation.
- **Fine-tuning (or transfer learning)**: The trained encoder is then fine-tuned with a prediction head for the supervised task dataset $\mathcal{D}$.

The choice feature encoder $h(\mathbf{x})$ can be arbitrary as long as it produces a fixed-length feature from tabular row. Therefore, various architectures can be employed. As described in Section 5.1, we

choose FT-Transformer as our main encoder due to its input encoding scheme (feature tokenizer). Alternatively, GNN-based encoders or simple fully-connected networks can be adapted depending on the dataset of interest.

Several learning objectives can be used for SSL (Johnson et al., 2023; Bardes et al., 2022):

- **Spectral contrastive loss**: As described above, this approach leverages spectral properties of the data.
- **NT-Xent/NT-Logistic**: These is a widely used class of objectives in contrastive learning that maximizes similarity between augmented views of the same instance while minimizing similarity with negative samples.
- **Non-contrastive losses**: One can factorize the positive-pair kernel using *non-contrastive* objectives, which extend the Rayleigh quotient to this stochastic setting (Horn and Johnson, 2012). A popular instance is the VICReg objective (Bardes et al., 2022), which is a Lagrangian objective towards solving:

$$\begin{aligned} \mathcal{L}_{\text{non-con}}(h) := \mathbb{E}\left\|h(\mathbf{X}^1) - h(\mathbf{X}^2)\right\|_2^2 \\ \text{s.t. } \text{Cov}(h(\mathbf{X}^i)) = I,\ i = 1, 2, \end{aligned} \tag{16}$$

  where the joint distribution over $\mathbf{X}^1$ and $\mathbf{X}^2$ is given by

$$\int \mathbb{Q}(\mathbf{X}^1 \mid \mathbf{x}) \cdot \mathbb{Q}(\mathbf{X}^2 \mid \mathbf{x}) \cdot d\mathbb{P}(\mathbf{x}),$$

  i.e., the distribution induced by first sampling $\mathbf{x}$ from the data distribution, then sampling $\mathbf{X}^1$ and $\mathbf{X}^2$ from the value transition distribution conditioned on $\overline{\mathbf{x}}$.

From several experiments, we found that the InfoNCE loss (Oord et al., 2018) which falls under NT-Xent, shows better performance than the spectral contrastive loss.

Similar to the benchmark method in Ye et al. (2024a), we perform hyper-parameter tuning to benchmark our methods. For the our self-supervised learning method in Section 4.4, the training of our knowledge-enriched model occurs in two steps, so we searched for hyper-parameters only in the second step. Please refer to Section E for details.

Appendix E. Hyper-parameter Search and Training Details

To optimize our methods to each dataset of KE-TALENT benchmark, we conducted hyper-parameter optimization using Optuna Akiba et al. (2019) with the Tree-structured Parzan Estimator (TPE) sampler, the default search strategy in Optuna, following Ye et al. (2024a). The search process aimed to optimize the root mean squared error (RMSE) for regression tasks and maximize the accuracy for classification tasks on the validation set. After finding the optimal hyper-parameters for each dataset and model, we trained the model 15 times with different random seeds and reported the test set performance. The search space included both architectural and training parameters as detailed in Table 5 for the hyper-parameter search space.

For the smoothing and value kernel models, which use RealMLP as their MLP architecture, we searched over the same hyper-parameter space as the TALENT benchmark, except for narrowing

the range of the learning rate to prevent instability in training. We used the AdamW optimizer with $\beta_1 = 0.9, \beta_2 = 0.95$, a batch size of 256, and a coslog$_4$ learning rate scheduler. In our implementation of RealMLP, we omitted data-driven weight initialization or a decaying dropout ratio due to the implementation complexity. Note that, despite these simplifications, our version remains closer to the original RealMLP compared to its simplified variant, RealMLP-TD-S.

For the concept graph attention network (CGAT) model, we used dataset-specific batch sizes. Specifically, we use the largest possible batch size (a power of two, up to 1024) for each dataset that could fit within 48GB of VRAM (NVIDIA RTX A6000 GPU).

For the self-supervised learning (SSL) model, hyper-parameter search was performed only during the fine-tuning stage, as described in Appendix D.2, focusing on the MLP prediction head and optimizer settings. During the SSL stage, we followed the TALENT codebase for default hyper-parameters of the encoder (FT-Transformer) architecture. We employed the AdamW optimizer with a cosine learning rate scheduler with warmup, where the base learning rate was set to 2×10^{-5}, and no weight decay. The model was trained with a batch size of 1024, and the number of epochs was chosen as the minimum multiple of 100 that ensured training for 1000 steps. The number of transition steps during data augmentation was adjusted per dataset to ensure that approximately 20% of columns were modified in each augmentation step. For the InfoNCE loss, we set the temperature parameter to $\tau = 0.1$.

Appendix F. Additional Results

F.1. Best result for each dataset

To provide a more complete picture, Table 6 lists extended KE-TALENT results, including the single best score for every dataset in the original TALENT benchmark. We did not treat these methods as primary baselines because (i) they were not the overall top performer within their respective model families, or (ii) their results became public only after our paper was written.

F.2. Concept kernel visualization

In Figures 2 to 4, we visualize the concept kernels for three datasets with 20–34 columns: Student Performance, German Credit Data, and Student Dropout. Additionally, Tables 7 to 9, lists the top-5 nearest columns (highest cosine similarity in the concept embedding space, excluding self) for each column in those datasets. Many of these neighbor pairs are indeed semantically related. For example:

- **Student Performance**
  - "number of school absences" and "number of past class failures"
  - "weekend alcohol consumption" and "workday alcohol consumption"
  - "extra educational support" and "family educational support"
  - "mother's education" and "father's education"
- **German Credit Data**
  - "Credit amount" and "Number of existing credits at this bank"
  - "Present residence since" and "Present employment since"

Methods	Parameters	Grid
Smoothing	Smoothing ξ (Only for norm and Laplacian)	LogUniform [0.1, 10.0]
	Numerical embedding	{none, pbld}
	Dropout	{0.0, 0.15}
	Nonlinear activation	{SELU, Mish}
	MLP hidden layers	{[256,256,256],[64,64,64,64,64],[512]}
	PLR sigma	LogUniform [0.05, 0.5]
	Label smoothing (only for classification)	{0.0, 0.1}
	Learning rate	LogUniform [0.02, 0.07]
	Weight decay	{0.0, 0.02}
Value kernel	Spectral decomposition matrix	{Adjacency, Laplacian}
	Numerical embedding	{none, pbld}
	Dropout	{0.0, 0.15}
	Nonlinear activation	{SELU, Mish}
	MLP hidden layers	{[256,256,256],[64,64,64,64,64],[512]}
	PLR sigma	LogUniform [0.05, 0.5]
	Label smoothing (only for classification)	{0.0, 0.1}
	Learning rate	LogUniform [0.02, 0.07]
	Weight decay	{0.0, 0.02}
Concept graph attention networks (CGAT)	Input embed dim	LogInt [16, 256]
	Conv num layers	UniformInt [1, 3]
	Conv hidden dim	LogInt [16, 64]
	Concept attention dim	LogInt [4, 16]
	Num attn heads	UniformInt [1, 4]
	Edge active ratio	Uniform [0.1, 0.9]
	MLP num layers	UniformInt [2, 5]
	MLP hidden dim	LogInt [16, 256]
	MLP dropout	0.1
	Learning rate	LogUniform [3e-5, 1e-3]
	Weight decay	LogUniform [1e-6, 1e-3]
SSL	MLP num layers	UniformInt [2, 5]
	MLP hidden dim	LogInt [16, 256]
	MLP dropout	0.1
	Learning rate	LogUniform [3e-5, 3e-3]
	Weight decay	LogUniform [1e-6, 1e-3]

Table 5: Hyper-parameter space for knowledge-enriched supervised learning methods

  - "Housing" and "Property"
  - "Other installment plans" and "Installment rate in percentage of disposable income"

- **Student Dropout**

Dataset	**Abalone**	**Diamond**	**ParkTel**	**StuPerf**	**Crime**	**Churn**	**Credit**	**Taiwan**	**ASP**	**Internet**	**StuDrop**
Method \ Task	reg/RMSE(↓)					bincls/Acc(↑)			multicls/Acc(↑)		
RealMLP	2.1210	523.92	**0.7337**	2.9277	0.1381	0.8735	0.7157	0.9667	0.3861	0.5302	0.7655
CatBoost	2.1789	524.91	1.5994	2.9244	0.1336	0.8759	0.7430	0.9718	0.3815	**0.5358**	**0.7782**
TabR	2.1078	513.53	8.0521	2.9072	0.1437	0.8743	0.7240	0.9678	0.3750	0.5183	0.7493
FT-T	2.1078	532.83	8.3437	2.9642	0.1369	0.8709	0.7123	0.9674	0.3678	0.5348	0.7547
Best	2.0888 (ResNet)	**492.58** (MNCA)		**2.5560** (MNCA)	**0.1325** (XGB)		**0.7513** (LGBM)		**0.4448** (XGB)		
Smooth(kernel)	2.1718	938.03	2.4700	3.0651	0.1466	0.8657	0.7160	**0.9722**	0.3815	0.5042	0.7162
Smooth(norm)	2.0879	903.70	1.2112	2.9725	0.1401	0.8688	0.6960	0.9659	0.4013	0.5093	0.7579
Smooth(Laplacian)	2.0937	522.37	0.9530	2.8926	0.1397	**0.8765**	0.7193	0.9694	0.3900	0.5259	0.7673
Value kernel	**2.0825**	525.79	0.8676	2.9203	0.1394	0.8746	0.7157	0.9673	0.3761	0.5315	0.7665
CGAT	2.0876	677.33	1.4612	3.0397	0.1425	0.8764	0.7337	0.9675	0.3838	0.5275	0.7656
SSL	2.1584	534.12	1.1275	2.9178	0.1373	0.8757	0.7180	0.9655	0.3964	0.5327	0.7571

Table 6: **Additional Results on KE-TALENT benchmark** Additionally to Table 2, the table reports the best method for each dataset in the "Best" row in case if the earlier four baselines didn't achieve the best in TALENT (MNCA: ModernNCA (Ye et al., 2024b), XGB: XGBoost (Chen and Guestrin, 2016), LGBM: LightGBM (Ke et al., 2017)).

- "Mother's qualification" and "Father's qualification"
- "Previous qualification" and "Previous qualification (grade)"
- "Nationality" and "International"
- "Mother's occupation" and "Father's occupation"

Because the concept embeddings were generated by a pretrained language model, these findings further demonstrate that modern LMs are effective at computing concept kernels that capture meaningful semantic relationships among dataset columns.

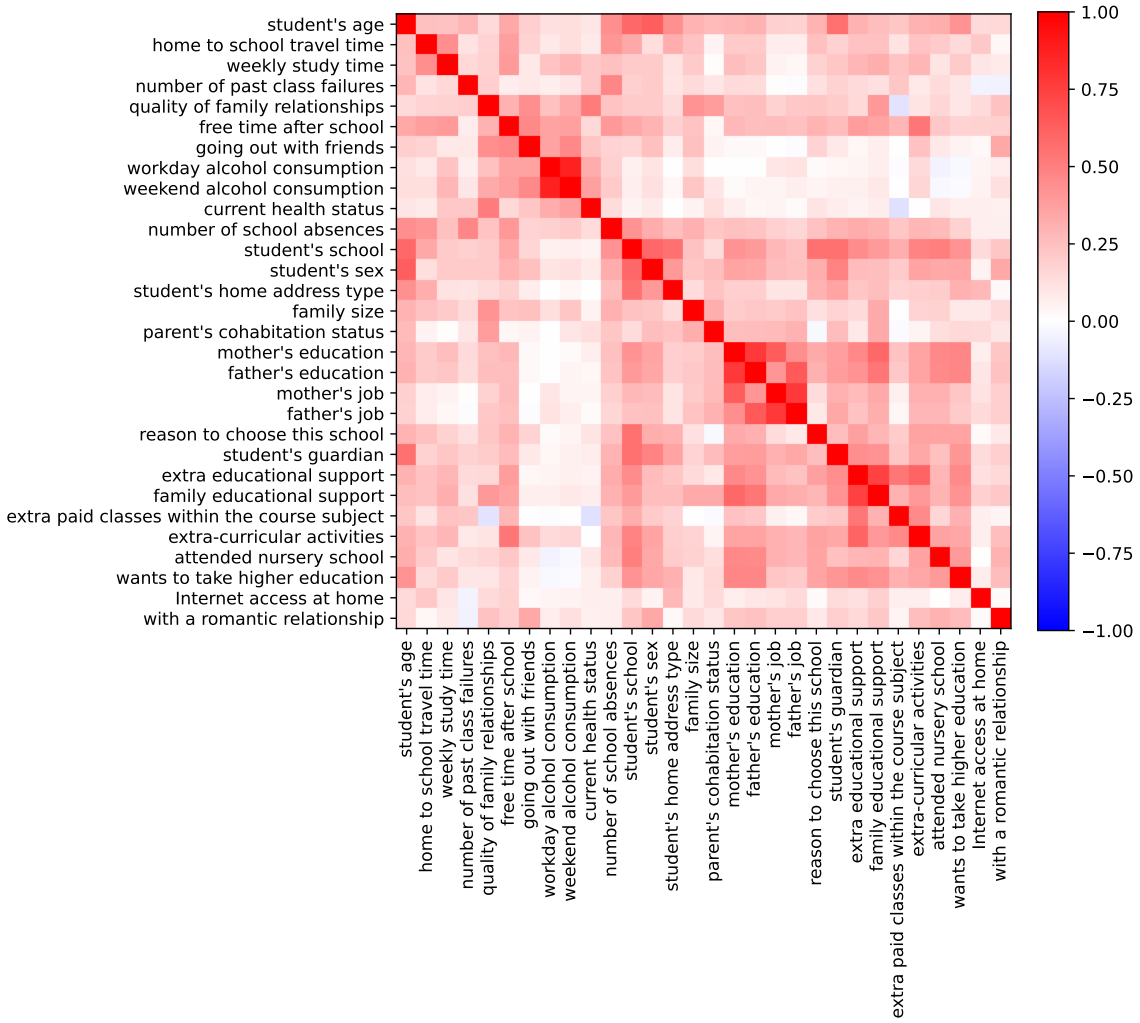

Figure 2: Kernel heatmap of Student Performance dataset

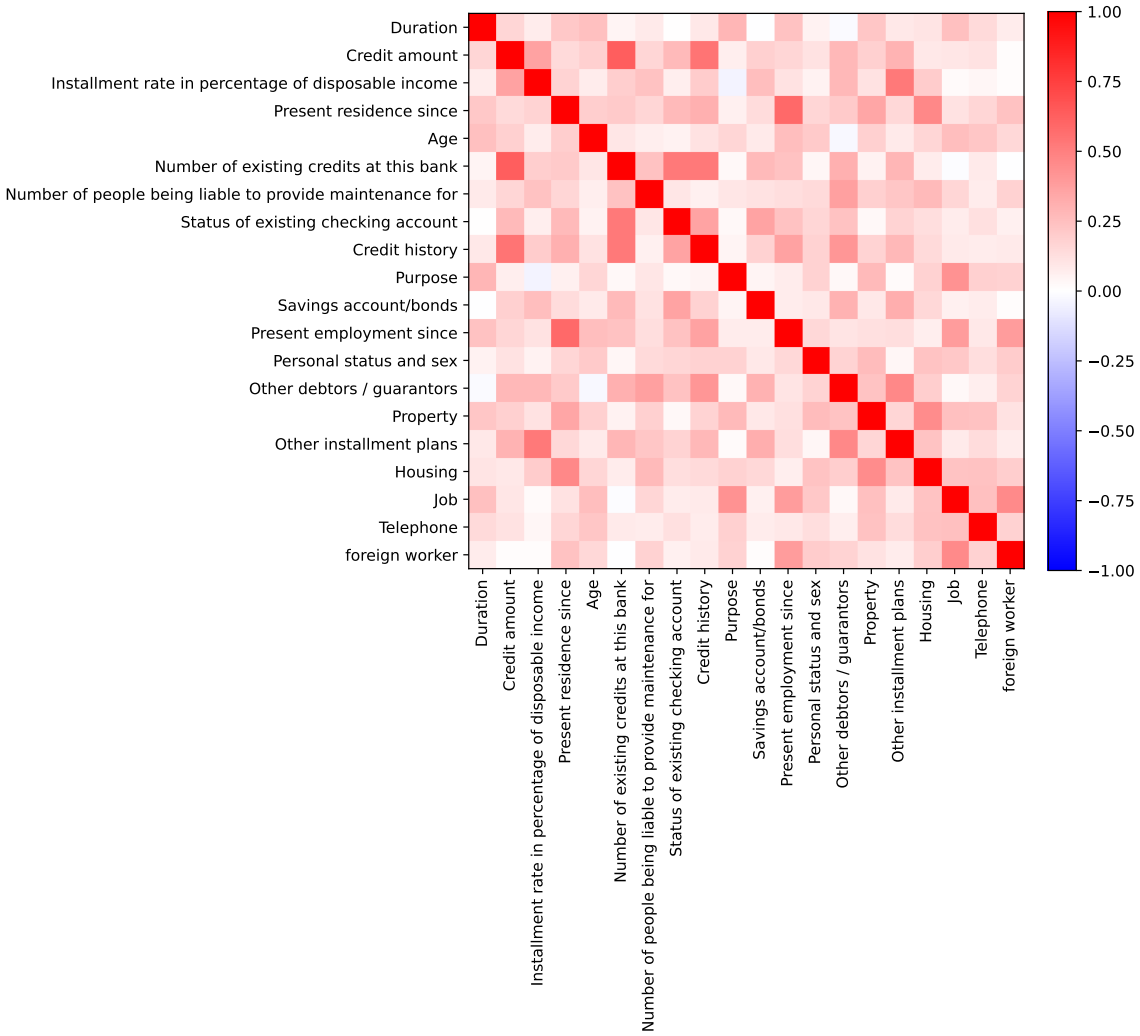

Figure 3: Kernel heatmap of German Credit Data dataset

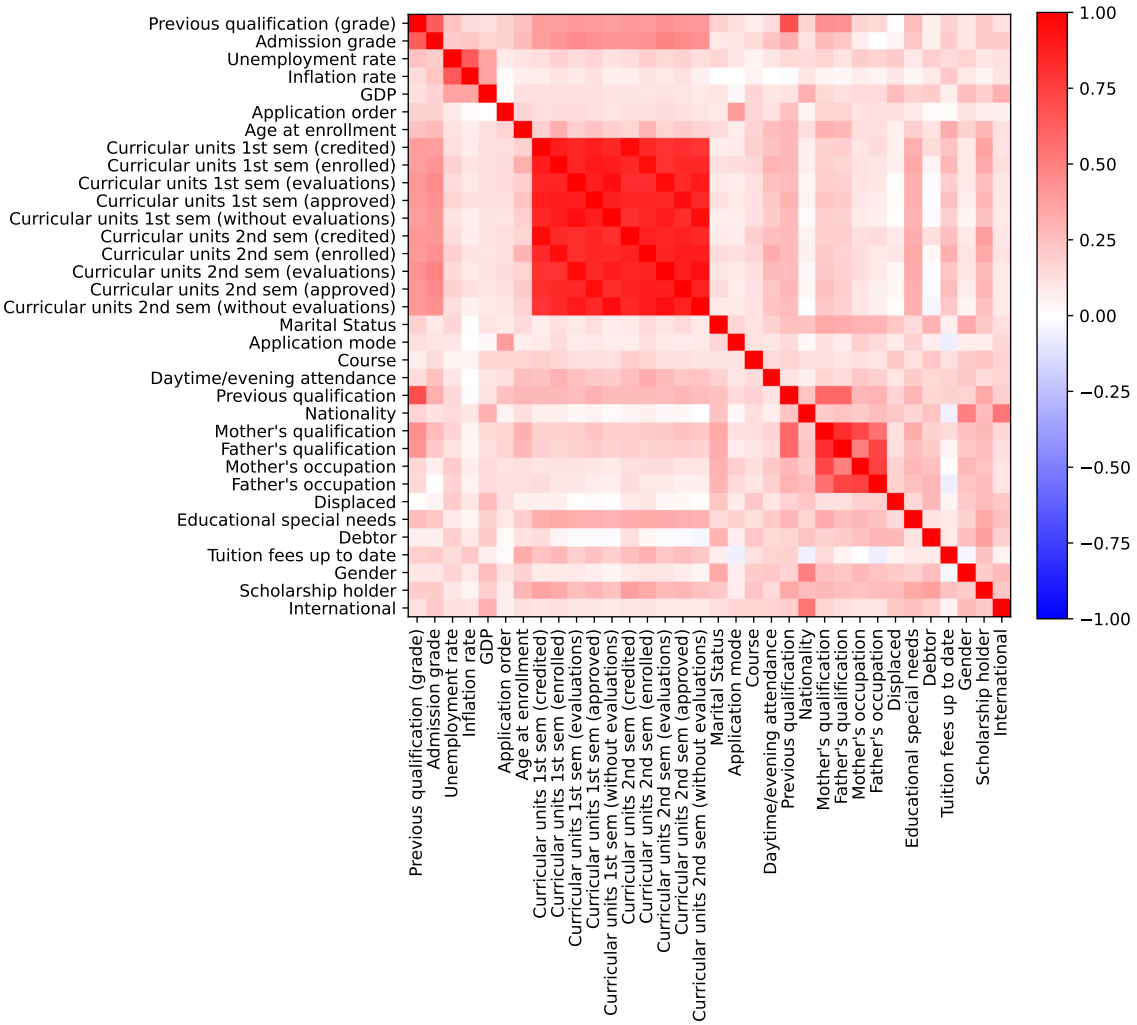

Figure 4: Kernel heatmap of Student Dropout dataset

Column	Top-5 Nearest Columns
"student's age"	"student's sex" "student's school" "student's guardian" "number of school absences" "student's home address type"
"home to school travel time"	"weekly study time" "number of school absences" "free time after school" "student's school" "student's home address type"
"weekly study time"	"home to school travel time" "free time after school" "family educational support" "extra-curricular activities" "extra educational support"
"number of past class failures"	"number of school absences" "student's age" "extra paid classes within the course subject" "current health status" "student's sex"
"quality of family relationships"	"current health status" "going out with friends" "family size" "family educational support" "parent's cohabitation status"
"free time after school"	"extra-curricular activities" "going out with friends" "number of school absences" "weekly study time" "extra educational support"
"going out with friends"	"weekend alcohol consumption" "free time after school" "quality of family relationships" "workday alcohol consumption" "with a romantic relationship"
"workday alcohol consumption"	"weekend alcohol consumption" "going out with friends" "free time after school" "current health status" "quality of family relationships"
"weekend alcohol consumption"	"workday alcohol consumption" "going out with friends" "current health status" "free time after school" "quality of family relationships"
"current health status"	"quality of family relationships" "weekend alcohol consumption" "workday alcohol consumption" "going out with friends" "weekly study time"
"number of school absences"	"number of past class failures" "student's age" "student's school" "home to school travel time" "free time after school"
"student's school"	"student's sex" "student's age" "student's guardian" "reason to choose this school" "student's home address type"
"student's sex"	"student's age" "student's school" "student's guardian" "student's home address type" "extra-curricular activities"
"student's home address type"	"student's school" "student's age" "student's sex" "student's guardian" "home to school travel time"
"family size"	"quality of family relationships" "family educational support" "parent's cohabitation status" "number of school absences" "student's age"
"parent's cohabitation status"	"quality of family relationships" "family educational support" "family size" "father's job" "student's age"
"mother's education"	"father's education" "mother's job" "family educational support" "wants to take higher education" "attended nursery school"
"father's education"	"mother's education" "father's job" "family educational support" "wants to take higher education" "attended nursery school"
"mother's job"	"father's job" "mother's education" "father's education" "family educational support" "student's guardian"
"father's job"	"mother's job" "father's education" "mother's education" "student's guardian" "family educational support"
"reason to choose this school"	"student's school" "extra educational support" "extra-curricular activities" "attended nursery school" "wants to take higher education"
"student's guardian"	"student's school" "student's age" "student's sex" "extra educational support" "family educational support"
"extra educational support"	"family educational support" "extra-curricular activities" "extra paid classes within the course subject" "mother's education" "wants to take higher education"
"family educational support"	"extra educational support" "mother's education" "father's education" "wants to take higher education" "student's guardian"
"extra paid classes within the course subject"	"extra educational support" "extra-curricular activities" "student's school" "wants to take higher education" "family educational support"
"extra-curricular activities"	"extra educational support" "free time after school" "student's school" "extra paid classes within the course subject" "family educational support"
"attended nursery school"	"student's school" "mother's education" "father's education" "wants to take higher education" "extra-curricular activities"
"wants to take higher education"	"mother's education" "father's education" "extra educational support" "student's age" "family educational support"
"Internet access at home"	"student's home address type" "home to school travel time" "family educational support" "free time after school" "student's school"
"with a romantic relationship"	"student's sex" "going out with friends" "attended nursery school" "wants to take higher education" "extra-curricular activities"

Table 7: **Top-5 nearest columns for each column in Student Performance dataset**

Column	Top-5 Nearest Columns
"Duration"	"Purpose" "Job" "Age" "Present employment since" "Property"
"Credit amount"	"Number of existing credits at this bank" "Credit history" "Installment rate in percentage of disposable income" "Other installment plans" "Other debtors / guarantors"
"Installment rate in percentage of disposable income"	"Other installment plans" "Credit amount" "Other debtors / guarantors" "Savings account/bonds" "Number of people being liable to provide maintenance for"
"Present residence since"	"Present employment since" "Housing" "Property" "Credit history" "Status of existing checking account"
"Age"	"Job" "Present employment since" "Duration" "Telephone" "Personal status and sex"
"Number of existing credits at this bank"	"Credit amount" "Credit history" "Status of existing checking account" "Other debtors / guarantors" "Other installment plans"

Column	Top-5 Nearest Columns
"Number of people being liable to provide maintenance for"	"Other debtors / guarantors" "Housing" "Number of existing credits at this bank" "Installment rate in percentage of disposable income" "Other installment plans"
"Status of existing checking account"	"Number of existing credits at this bank" "Savings account/bonds" "Credit history" "Credit amount" "Present residence since"
"Credit history"	"Credit amount" "Number of existing credits at this bank" "Other debtors / guarantors" "Present employment since" "Status of existing checking account"
"Purpose"	"Job" "Duration" "Property" "Telephone" "foreign worker"
"Savings account/bonds"	"Status of existing checking account" "Other installment plans" "Other debtors / guarantors" "Number of existing credits at this bank" "Installment rate in percentage of disposable income"
"Present employment since"	"Present residence since" "foreign worker" "Job" "Credit history" "Age"
"Personal status and sex"	"Property" "Housing" "Job" "Age" "foreign worker"

Column	Top-5 Nearest Columns
"Other debtors / guarantors"	"Other installment plans" "Credit history" "Number of people being liable to provide maintenance for" "Number of existing credits at this bank" "Savings account/bonds"
"Property"	"Housing" "Present residence since" "Purpose" "Personal status and sex" "Job"
"Other installment plans"	"Installment rate in percentage of disposable income" "Other debtors / guarantors" "Savings account/bonds" "Credit amount" "Number of existing credits at this bank"
"Housing"	"Present residence since" "Property" "Number of people being liable to provide maintenance for" "Telephone" "Personal status and sex"
"Job"	"foreign worker" "Purpose" "Present employment since" "Age" "Duration"
"Telephone"	"Job" "Housing" "Property" "Age" "Purpose"
"foreign worker"	"Job" "Present employment since" "Present residence since" "Personal status and sex" "Housing"

Table 8: **Top-5 nearest columns for each column in German Credit Data dataset**

Column	Top-5 Nearest Columns
"Previous qualification (grade)"	"Previous qualification" "Admission grade" "Mother's qualification" "Father's qualification" "Curricular units 2nd sem (evaluations)"
"Admission grade"	"Previous qualification (grade)" "Curricular units 2nd sem (evaluations)" "Curricular units 1st sem (evaluations)" "Curricular units 2nd sem (approved)" "Curricular units 2nd sem (without evaluations)"
"Unemployment rate"	"Inflation rate" "GDP" "Previous qualification (grade)" "Mother's occupation" "Admission grade"
"Inflation rate"	"Unemployment rate" "GDP" "Admission grade" "Tuition fees up to date" "Previous qualification (grade)"
"GDP"	"Unemployment rate" "Inflation rate" "Nationality" "International" "Displaced"
"Application order"	"Application mode" "Previous qualification" "Previous qualification (grade)" "Admission grade" "Father's qualification"
"Age at enrollment"	"Tuition fees up to date" "Curricular units 1st sem (enrolled)" "Mother's qualification" "Curricular units 2nd sem (enrolled)" "Previous qualification"
"Curricular units 1st sem (credited)"	"Curricular units 2nd sem (credited)" "Curricular units 1st sem (enrolled)" "Curricular units 1st sem (approved)" "Curricular units 1st sem (evaluations)" "Curricular units 1st sem (without evaluations)"
...	

Column	Top-5 Nearest Columns
...	
"Curricular units 2nd sem (without evaluations)"	"Curricular units 1st sem (without evaluations)" "Curricular units 2nd sem (evaluations)" "Curricular units 1st sem (evaluations)" "Curricular units 2nd sem (enrolled)" "Curricular units 2nd sem (approved)"
"Marital Status"	"Mother's qualification" "Gender" "Father's qualification" "Mother's occupation" "Debtor"
"Application mode"	"Application order" "Mother's occupation" "Educational special needs" "International" "Marital Status"
"Course"	"Scholarship holder" "Debtor" "Displaced" "Gender" "Curricular units 2nd sem (credited)"
"Daytime/evening attendance"	"Curricular units 2nd sem (enrolled)" "Curricular units 1st sem (enrolled)" "Curricular units 2nd sem (evaluations)" "Curricular units 2nd sem (credited)" "Age at enrollment"
"Previous qualification"	"Previous qualification (grade)" "Mother's qualification" "Father's qualification" "Scholarship holder" "Admission grade"
"Nationality"	"International" "Gender" "GDP" "Scholarship holder" "Father's occupation"
"Mother's qualification"	"Father's qualification" "Mother's occupation" "Previous qualification" "Father's occupation" "Previous qualification (grade)"
"Father's qualification"	"Mother's qualification" "Father's occupation" "Previous qualification" "Mother's occupation" "Previous qualification (grade)"

Column	Top-5 Nearest Columns
"Mother's occupation"	"Father's occupation" "Mother's qualification" "Father's qualification" "Marital Status" "Previous qualification"
"Father's occupation"	"Mother's occupation" "Father's qualification" "Mother's qualification" "Previous qualification" "Marital Status"
"Displaced"	"Debtor" "Scholarship holder" "GDP" "Marital Status" "Nationality"
"Educational special needs"	"Curricular units 2nd sem (enrolled)" "Scholarship holder" "Curricular units 1st sem (enrolled)" "Curricular units 2nd sem (evaluations)" "Mother's qualification"
"Debtor"	"Scholarship holder" "Marital Status" "Father's occupation" "Displaced" "Gender"
"Tuition fees up to date"	"Age at enrollment" "Curricular units 2nd sem (enrolled)" "Curricular units 1st sem (enrolled)" "Curricular units 2nd sem (approved)" "Curricular units 2nd sem (credited)"
"Gender"	"Nationality" "Marital Status" "Mother's occupation" "International" "Debtor"
"Scholarship holder"	"Curricular units 2nd sem (credited)" "Debtor" "Curricular units 1st sem (credited)" "Curricular units 2nd sem (enrolled)" "Educational special needs"
"International"	"Nationality" "GDP" "Gender" "Educational special needs" "Displaced"

Table 9: **Top-5 nearest columns for each column in Student Dropout dataset** We omitted several redundant columns (from "Curricular units 1st sem (enrolled)" to "Curricular units 2nd sem (approved)") due to space constraints.