Semantic-based Data Augmentation for Machine Learning Prediction Enhancement

Majlinda Llugiqi [a,*], Fajar J. Ekaputra [a] and Marta Sabou [a]
[a] *Institute for Data, Process and Knowledge Management, Vienna University of Economics and Business, Vienna, Austria*
E-mails: majlinda.llugiqi@wu.ac.at, fajar.ekaptura@wu.ac.at, marta.sabou@wu.ac.at

Abstract.
Machine learning (ML) methods have demonstrated strong predictive capabilities when trained on large datasets. However, in domains where data is scarce or sensitive, ML models often exhibit suboptimal performance. Our hypothesis is that semantically enriching the available training dataset can enhance the predictive power of ML models, particularly in data-scarce scenarios. To investigate this hypothesis, we propose novel neuro-symbolic approaches that augment tabular data with KG information, providing additional context and structure to improve model performance. Concretely, we introduce and examine several integration techniques of KG information through embeddings and explore how different KG embedding algorithms affect model performance, with a specific focus on accuracy and F2 scores. Our evaluation involves four distinct ML algorithms and four KG embedding techniques. We apply our approach to binary classification tasks on tabular data, including heart disease and chronic kidney disease. Our experimental results show improvements in performance particularly when tabular data is augmented with distance features computed in the embedding space. Notably, we achieve gains in F2 scores, such as an increase in XGBoost performance from 75.19% to 90.85% for heart disease prediction. These findings demonstrate the potential of KG-based augmentation to enhance ML performance.

Keywords: Neuro-symbolic AI, Knowledge Graph Embeddings, Machine Learning, Data Augmentation

1. Introduction

Machine learning (ML) has revolutionized various domains by providing powerful tools for pattern recognition, predictive analytics, and data-driven decision-making. Techniques such as deep learning have achieved remarkable success in fields ranging from computer vision [12, 51] to natural language processing [29, 38]. These advancements have been largely driven by the availability of large datasets and the computational power to process them.

However, ML methods often face significant challenges related to data quality and availability. Data sparsity, imbalance, and sensitivity can severely hinder the performance of ML models [36]. In the medical domain, one important task is predicting patient outcomes, for instance, determining the presence or absence of a disease based on clinical observations. This task often suffers from an insufficient amount of labeled data due to privacy concerns [25]. Although advances have been made, models trained solely on tabular data fail to fully capture the domain's complexity and semantics, limiting their ability to generalize effectively [41].

To overcome these limitations, neuro-symbolic (NeSy) AI has emerged as a promising approach to integrate domain knowledge into ML models. NeSy AI combines the strengths of symbolic AI—known for logical reasoning

*Corresponding author. E-mail: majlinda.llugiqi@wu.ac.at.

and explainability—with sub-symbolic methods such as deep learning [14, 21, 42]. In particular, structured semantic knowledge such as knowledge graphs (KGs) has emerged as a key element in bridging the gap, providing a structured way to represent relationships between entities and capture domain-specific semantics [4, 15, 20, 55]. KGs have been widely used in tasks such as knowledge graph completion [31] and link prediction [49]. However, their potential to enhance ML predictions on tabular data by incorporating semantic knowledge through embeddings remains underexplored.

We propose integrating KGs into ML pipelines to enhance tabular data with structured, domain-specific information. Drawing upon techniques from the Semantic Web community, our approach begins by utilizing ontologies to formalize domain semantics. We then construct KGs based on these ontologies, enriching the datasets with structured knowledge specific to the medical domain. Subsequently, we employ knowledge graph embeddings to transform the KGs into numerical vector representations suitable for ML algorithms. By embedding relationships and domain knowledge from KGs into these vectors, our methodology enhances the ML pipeline by augmenting the datasets with semantic knowledge, aiming to improve predictive performance—especially in data-scarce domains. This study specifically explores binary classification tasks in both medical predictions (heart disease and chronic kidney disease) where domain-specific structure is crucial for robust prediction. Our research is guided by the following research questions:

- RQ1: How can KGs be optimally infused into an ML pipeline to enhance performance in terms of accuracy and F2 score?
- RQ2: How does the choice of knowledge graph embedding algorithms affect the performance of machine learning models when used to augment tabular data?
- RQ3: How do different ML algorithms perform when KG-based information is integrated into the input data?

To address these research questions, we took an exploratory approach, systematically investigating each aspect step by step. For RQ1, we derived five sub-hypotheses to examine how knowledge graphs can be optimally integrated into ML pipelines to enhance performance metrics like accuracy and F2 score. We tested these hypotheses using eight different approaches, each incorporating knowledge graphs and embeddings in various ways. For RQ2 and RQ3, we empirically evaluated the impact of different knowledge graph embedding algorithms and ML models across two medical domains—heart disease and chronic kidney disease prediction.

Building on our previous work [32], we extend and formalize our methodology for integrating KG embeddings into ML pipelines. We employ two additional embedding techniques alongside those used previously to transform the KGs into numerical vector representations suitable for ML algorithms. We developed and tested different approaches based on five sub-hypotheses derived from our first research question, providing a comprehensive evaluation of their impact on model performance in heart and kidney disease prediction. Our study demonstrates the effectiveness of incorporating ontological knowledge into the ML training process, highlighting the potential for improved predictive performance in data-scarce domains and its applicability across various fields where ontologies can be developed or expanded.

The remainder of this paper is organized as follows: In Section 2, we define the key concepts that we use in our work. This is followed by an overview of related work in Section 3. In Section 4, we present an overview of our proposed approach, with more detailed explanations about our approaches provided in Section 5. Our experimental analysis is discussed in Section 6, where we outline the goals and setup of our experiments. In Section 7, we present and analyze the outcomes of our experiments. Finally, we summarize our findings and outline directions for future work in Section 8.

2. Problem Description and Background Information

In this section, we outline the problem we aim to address, followed by introducing the key concepts that we use throughout the paper, beginning with ontologies, knowledge graphs, and knowledge graph embeddings.

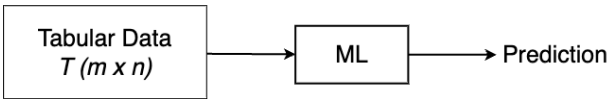

Fig. 1. Baseline for ML prediction on tabular data.

2.1. Problem Description

In this study, we address the challenge of predicting heart disease and chronic kidney disease using patient medical records in tabular data format. Each dataset can be represented as a table $T \in \mathbb{R}^{n \times m}$, where n is the number of patient instances and m is the number of features or attributes. These features capture patient demographics, clinical measurements, and diagnostic information essential for disease prediction. For heart disease prediction, we consider features such as age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, and resting electrocardiogram results. The kidney disease dataset similarly includes essential attributes, including age, blood pressure, specific gravity, albumin, blood glucose, blood urea, serum creatinine, hemoglobin, and red and white blood cell counts.

We focus on binary classification to predict the presence or absence of these diseases. Formally, given the dataset T, the goal is to learn a function $f : \mathbb{R}^m \rightarrow \{0, 1\}$ that maps a patient's feature vector to a binary outcome indicating disease presence (1) or absence (0). As illustrated in Figure 1, the tabular data T serves as input to machine learning models, which then output predictions regarding disease presence.

For example, given a patient's data (e.g., age 62, female, asymptomatic, resting blood pressure 140, cholesterol 268, no fasting blood sugar, max heart rate 160, downsloping slope and thalassemia), our model aims to determine the likelihood of heart disease. Similarly, a record for kidney disease might involve attributes such as age 68, blood pressure 70, specific gravity 1.01, and blood urea 54. The objective is to accurately predict disease presence.

Due to the sensitive nature of medical data, datasets in this domain are often limited or partially incomplete, impacting model performance. This scarcity of data, combined with varying data quality, presents a challenge to achieving optimal prediction accuracy, necessitating robust preprocessing and, potentially, data augmentation strategies to improve model generalizability and reliability.

2.2. Background Information

Given the problem definition described in the previous subsection, our approach aims to augment these datasets by integrating semantic information to enhance predictive capabilities. To achieve this, we leverage *ontologies* to capture the domain knowledge, and then we use *knowledge graphs* to enrich the datasets with ontologies. We then need *knowledge graph embeddings* to transform the knowledge graphs into a vector space suitable for machine learning models. In the following we discuss each of these concepts in detail.

Ontology: Originally a philosophical term, ontology refers to the study of existence and the nature of being. In computer science, Gruber [18] redefined ontology as "explicit specifications of conceptualizations", where a conceptualization represents a simplified, abstract view of a domain to capture essential aspects. An ontology establishes a standardized vocabulary for knowledge sharing within a specific domain. Formally, an ontology represented as $O = (C, R, H^C)$ encompasses a collection of concepts C, a set of relations R, and a hierarchical structure of concepts H^C. Each relation $r \in R$ indicates an association between pairs of concepts, such that $r \subseteq C \times C$. The concept hierarchy H^C is a subset of $C \times C$, illustrating the relationships among concepts.

Knowledge Graphs: Knowledge graphs (KGs) expand on ontologies by capturing not only the structured relationships between concepts but also the specific instances and values within a domain. Originally popularized by Google in 2012 [45] to enhance search understanding, KGs have since become integral in a range of applications, providing a structured, machine-readable format to represent knowledge. We define the KG as $KG = (E, R', Tr)$ where:

- E represents the set of entities in the knowledge graph. Each entity $e \in E$ can represent a real-world concept, object or idea, such as 'Person' or 'City'.

- R' represents the set of instantiated relations between entities within the KG such as 'hasAge' or 'worksAt'.
- Tr denotes a set of triples, where each triple $tr = (e_1, e, e_2) \in Tr$ represents a fact or statement in the knowledge graph.

KG embeddings: While KGs provide a structured representation of entities and their relationships, they can become highly complex as the number of entities and relations grows. To enable efficient computation, learning, and reasoning over KGs, knowledge graph embeddings (KGEs) are commonly used [5, 31, 50]. KG embeddings transform entities and relations from a discrete symbolic space into a continuous vector space, capturing the structure and semantics of the KG in a form that is compatible with ML algorithms. KGE algorithms can be broadly categorized into three main types based on their methodology and objectives: translational distance models, semantic matching models, and random walk-based models. In the following, we briefly describe the embedding algorithms used in our experiments: Node2Vec [17] and Rdf2Vec [40] as random-walk based models that leverage the graph structure, DistMult [53] as a semantic matching model, and TransH [50] as a translational model.

- Node2Vec uses a flexible random walk strategy to combine depth-first and breadth-first sampling, allowing it to capture various structural features of the graph whether they are labeled or unlabeled, directed or undirected. Node2Vec employs random walks, incorporating an adjustable bias parameter that allows for targeted exploration of local neighborhoods as well as a broader global search.
- RDF2Vec is designed specifically for RDF (Resource Description Framework) graphs within the Semantic Web, RDF2Vec generates embeddings for entities and relations by leveraging random walks to create sequences from the graph. These sequences are then transformed into embeddings using Word2Vec, making RDF2Vec particularly effective at capturing the semantic and relational attributes present in RDF data. While both RDF2Vec and Node2Vec utilize random walks, RDF2Vec focuses more on semantic relationships within the context of the Semantic Web, whereas Node2Vec emphasizes structural characteristics applicable to a wider range of graph types.
- DistMult is a semantic matching model that uses a bilinear scoring function to evaluate the interactions between entities and relations in a knowledge graph. In this model, each relation is represented as a diagonal matrix, simplifying the bilinear form to a weighted element-wise multiplication of entity embeddings. While this approach effectively captures pairwise relationships, it inherently assumes that all relations are symmetric, which may restrict its expressiveness for datasets containing asymmetric relations.
- TransH is a translational model that represents entities as vectors and relations as hyperplanes in the embedding space. Each relation is associated with a specific hyperplane and a translation vector on that hyperplane. Entities are projected onto the hyperplane of a relation before the translation operation is applied. This method allows entities to have different representations in the context of different relations, enabling the model to capture complex and diverse relationships thereby improving its ability to represent multiple types of relationships in a knowledge graph.

3. Related work

We review related work on (i) the categorization of neuro-symbolic approaches, positioning our work within these categories, (ii) we discuss the use of ML models in disease prediction and (ii) enhancing ML predictions with semantic knowledge, and we conclude by discussing the novelty of our approach.

Categorization of Neuro-Symbolic Approaches In recent years, the field of neuro-symbolic AI has gained significant attention due to its potential to combine the strengths of both symbolic and sub-symbolic AI [14, 21, 42]. Symbolic AI excels at logical reasoning and explainability, while sub-symbolic approaches, such as deep learning, have proven effective in pattern recognition and data-driven decision-making. Combining these approaches, neuro-symbolic AI seeks to leverage the best of both worlds: the learning capability of sub-symbolic methods and the structured, interpretable reasoning of symbolic methods. Several efforts have focused on categorizing neuro-symbolic approaches. Kautz et al. [28] classify neurosymbolic systems into six types based on the interaction between neural networks and symbolic reasoning. *Type 1* employs standard deep learning with symbolic inputs and

outputs, while *Type 2* combines neural networks with symbolic solvers, as seen in systems such as AlphaGo. *Type 3* uses neural networks for tasks such as object detection, while symbolic systems handle complementary tasks such as query answering. In *Type 4*, symbolic knowledge is embedded into neural network training, whereas *Type 5* incorporates symbolic rules as constraints in the loss function. Finally, *Type 6* aims for fully integrated systems, merging symbolic reasoning with neural architectures, although fully mature combinatorial reasoning within such systems remains a challenge. Our approach belongs to *Type 4* of Kautz's classification, where symbolic knowledge is incorporated into the training process.

Similarly, Sheth et al. [30, 44] identify three levels of knowledge infusion in neural models: shallow, semi-deep, and deep. *Shallow infusion* introduces syntactic and symbolic knowledge at the input level, *semi-deep infusion* introduces external knowledge into intermediate layers via attention mechanisms or constraints, and *deep infusion* embeds structured, multi-layered knowledge into the network itself, aligning abstraction layers with learning stages. Our work adopts the *shallow infusion* approach by enriching input data with syntactic and symbolic knowledge, enhancing the model's performance.

Dash et al. [11] categorize methods for integrating domain-specific knowledge into deep neural networks into three main approaches: enhancing input data, modifying the loss function, and adjusting the network architecture. Our research aligns with the *input transformation* category, where domain-specific knowledge is integrated by enriching the input data provided to the ML models.

Van Harmelen and ten Teije [47] introduced a conceptual framework known as "boxology," which outlines various patterns for integrating machine learning with semantic web technologies. Breit et al. [6] expanded this framework by identifying 44 distinct patterns used in hybrid learning and reasoning techniques, based on a review of around 500 papers from 2010 to 2020. Our approach falls under the *T* patterns, specifically *T4*, where input transformations using symbolic knowledge are applied to improve model performance.

As a summary, our approach falls under the *shallow infusion* category as described by Sheth et al. [44], where syntactic and symbolic knowledge is introduced at the input level. It aligns with *Type 4* in Kautz's classification [28], as symbolic knowledge is embedded into the training process. Furthermore, it belongs to the *input transformation* approach discussed by Dash et al. [11], where domain-specific knowledge enhances the input data provided to machine learning models. Finally, our work corresponds to the *T4* pattern in the "boxology" framework, focusing on input transformations to improve model performance.

Machine Learning Models in Disease Prediction The application of ML in healthcare, particularly for disease prediction, has attracted significant research interest. For example, ML algorithms have been successfully employed in predicting diseases such as heart disease [27, 39, 43, 52] and kidney disease [8, 37, 48, 54], utilizing various techniques such as data preprocessing, feature selection, and hyperparameter tuning to enhance prediction accuracy.

Several studies have also explored combining ML methods to further improve performance. For instance, Mohan et al. [33] combined random forest and linear methods to enhance heart disease prediction, while Ali et al. [2] introduced a framework for heart failure prediction using dual support vector machine (SVM) models—one for feature selection and the other for the prediction task.

Although these models demonstrate good predictive performance, they often rely on extensive preprocessing [19], feature selection, and hyperparameter tuning to achieve optimal results. Moreover, the effectiveness of ML models can be limited by insufficient or sub-optimal quality data. In this context, healthcare ontologies [9, 13, 24, 26, 35] offer a structured, semantically rich layer of information that can enhance the contextual understanding of ML models, which is further explored in this work.

Enhancing ML Predictions with Semantic Knowledge Recent research has increasingly focused on integrating semantic knowledge, such as KGs and ontologies, into ML models to enhance their performance. KGs have been widely applied in various domains, notably improving feature extraction and entity representation in natural language processing tasks. For instance, Moussallem et al. [34] demonstrated how augmenting neural machine translation systems with KGs improved the translation quality by enhancing the semantic understanding of terminological expressions. Similarly, KG-based input enhancement has been shown to improve recommendation systems and community detection, enhancing both accuracy and explainability [4].

Moreover, KG-augmented neural networks have demonstrated improved performance in text classification and natural language inference tasks. Annervaz et al. [3] showed that integrating structured knowledge from KGs not

only improved model accuracy but also allowed models to perform well with less labeled data, addressing the common issue of data sparsity. Ziegler et al. [56] adopted a similar approach by incorporating semantic knowledge through graph embeddings for credit card fraud detection, demonstrating how the injection of background knowledge—such as public holidays from DBpedia—into neural models could enhance classification outcomes.

In addition to NLP and fraud detection, Szilagyi et al. [46] applied semantic knowledge in smart building management by integrating taxonomies, schemas, and logic rules with ML models. This hybrid system optimized building management by combining data-driven insights with rule-based reasoning, showing the potential of semantic knowledge in enhancing decision-making processes. Huang et al. [22] introduced an Abductive Learning with KG approach that automatically mines logic rules from KGs and integrates them into ML models using a knowledge-forgetting mechanism to filter irrelevant information, thereby improving model performance even with limited labeled data.

In healthcare, Gazzotti et al. [16] demonstrated how augmenting sparse electronic medical records (EMRs) with ontological resources improved the predictive capabilities of ML algorithms, specifically in hospitalization prediction. Ontologies such as SNOMED and UMLS provide structured medical knowledge, enabling a richer representation of patient data. Similarly, Ruiz et al. [41] introduced the PLATO method, which uses a KG to regularize a multilayer perceptron for tabular datasets, showing that semantic knowledge can help ML models handle high-dimensional and low-sample-size data more effectively.

These studies underscore the growing importance of integrating semantic knowledge into ML models to address challenges such as data quality, sparsity, and explainability across different domains.

Building on our previous work [32], we introduce a novel approach that leverages KGs to augment the tabular data by incorporating KG embeddings into ML models, a technique not yet explored in the context of medical prediction nor NeSy systems. We thoroughly investigate this integration, exploring several approaches to combining KG embeddings with ML, and calculating various metrics in the embedding space to enrich tabular data with additional semantic context. Additionally, we employ two additional techniques for generating embeddings and apply these methods to the prediction of heart disease and chronic kidney disease.

4. Knowledge Graph Embedding-Based Augmentation for Tabular Data

To improve the performance of ML models, we leverage KGs to enrich tabular datasets with semantic information. This section outlines the two core steps of our approach. First, in Section 4.1, we explain the creation of KGs using instances from the tabular data, as shown in the top part of Figure 2. This step focuses on building KGs that capture deeper relationships within the data. Next, in Section 4.2, we focus on integrating KG embeddings into the ML pipeline. This includes various augmentation strategies designed to enhance model performance by incorporating structural and relational information from the KG into the training data, as depicted in the bottom part of Figure 2.

4.1. Knowledge Graph Creation

For our approach to enrich ML input data with supplementary knowledge, constructing KGs is essential. They serve as structured representations of domain knowledge, capturing the semantics of the data and allowing for the integration of ontological information into datasets. This enrichment allows ML models to leverage contextual and relational information, enhancing their predictive capabilities. The upper part of Figure 2 illustrates the methodology used for building these KGs, which represent data that was initially captured in tabular form. The following steps provide a formal description of this construction process.

Step 1: Ontology Definition The first step in constructing the KG is defining an ontology, which is used to capture domain semantics and provide a structured framework for enriching the datasets. There are different ways to develop an ontology, represented as $O = (C, R, H^C)$. We considered (i) creating a new ontology from scratch, (ii) extending and reusing existing ontologies to include additional domain-specific information, or (iii) extracting relevant components from a more extensive ontology (see Section 6.2).

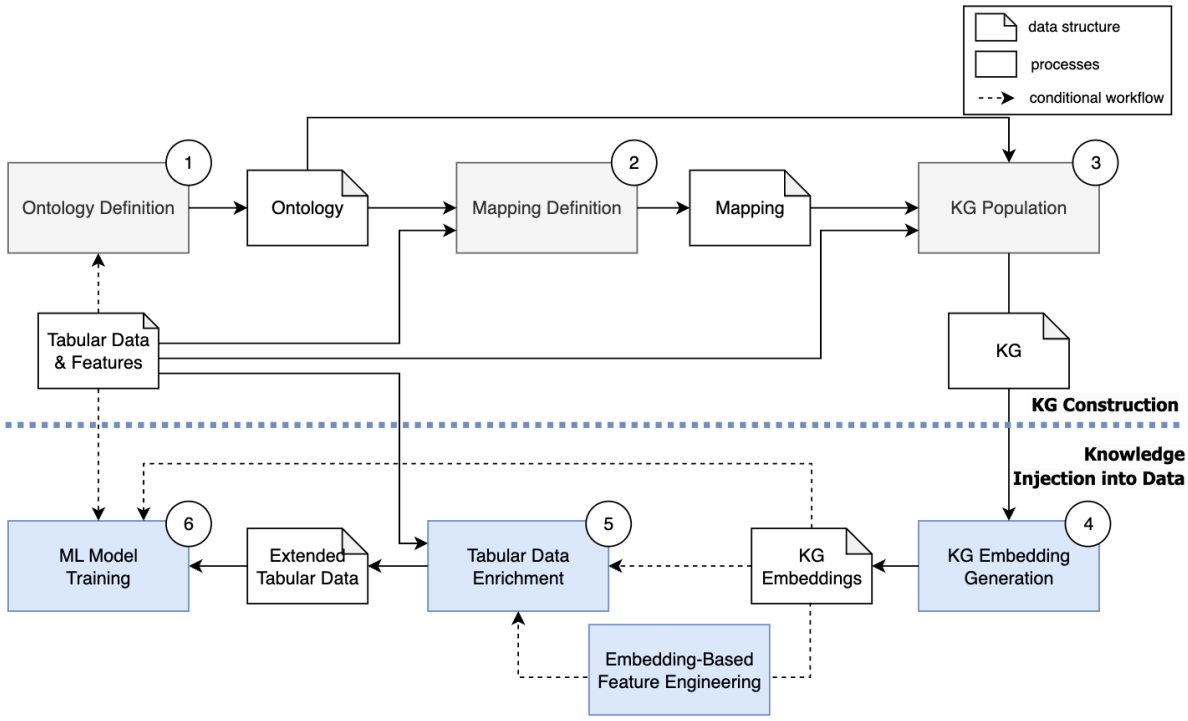

Fig. 2. Overview of the proposed approach including (i) KG construction (top) and (ii) knowledge injection into data (bottom) (adapted from [32]).

Step 2: Mapping Definition The process of mapping dataset features to the concepts in the ontology is crucial for using instances from tabular data to populate the ontology and, consequently, construct a KG. A key aspect of this mapping process is the **mapping function** $\psi : F \rightarrow C$, where $F = \{f_1, f_2, \ldots, f_n\}$ represents the features within the tabular dataset T, defined as a matrix of dimensions $m \times n$. This function entails the manual mapping of each feature f_i with a corresponding concept C in the ontology O.

Step 3: Knowledge Graph Population The knowledge graph KG is constructed by utilizing the ontology O along with the instances from the tabular data T and applying the mapping function $\psi : F \rightarrow C$. This process is automated through a Python script. We define the KG as $KG = (E, R', Tr)$ where:

- E signifies the set of entities, with each entity $e_i \in E$ corresponding to an instance in the tabular data derived from each row m_i in T,
- R' represents the set of instantiated relations within the KG, which includes relations from R through the mapping ψ, and illustrates direct relationships between entities E or between an entity and a literal value,
- Tr consists of triples generated for each feature value in an instance row m_i, following the mapping ψ. For instance, if a feature $f_{\text{glucoseLevel}}$ corresponds to an instance e_i with a blood pressure value of 95, the associated triple would be $(e_i, r_{\text{hasGlucoseLevel}}, 95)$, indicating the relationship $r_{\text{hasGlucoseLevel}}$ between entity e_i and the literal value 95.

This preprocessing phase ensures that features from the tabular data T are semantically represented within the Knowledge Graph KG using the defined ontology O.

4.2. Integrating KG Embeddings into ML Pipeline

In Section 4.1, we outlined the construction of enriched data structures that capture deeper semantics beyond the raw data. This section will now focus on transforming these enriched structures into a vectorized format suitable for ML, and on the optimal strategies for augmenting the input data, as illustrated in the lower part of Figure 2.

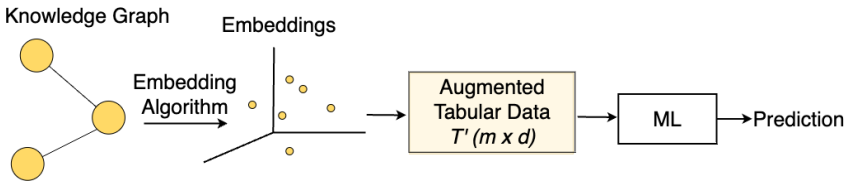

Fig. 3. Embeddings as ML model inputs.

Step 4: Knowledge Graph Embedding Generation With the populated KG with enriched data structures, the subsequent step is to prepare the KG for ML model training. This requires transforming the KG into a vector space representation suitable for ML models, using knowledge graph embedding (KGE) algorithms. Having a knowledge graph $KG = (E, R)$, the goal of the embedding algorithm is to map entities E and relations R into a continuous vector space. Formally, this can be represented as function: $\phi : E \cup R \rightarrow \mathbb{R}^d$, where ϕ is the embedding function that maps each entity and relation in the KG to a d -dimensional real-valued vector in the vector space $\mathbb{R}^d$. This transformation allows the KG to be represented in a way that preserves its semantic information while being computationally efficient. In the next steps 5 & 6 we will see how these embeddings are used as such or to compute features that are added to augment the dataset for a better ML performance.

Step 5 & 6: Tabular Data Enrichment and ML Model Training After computing KGEs, our objective is to explore the integration of these embeddings to enhance the performance of ML models. We experimented with different approaches for augmenting the training set using KGEs. First, we established a baseline that trains ML models using only tabular data T, following the traditional approach, shown in Figure 1, where no KG information is being added. Then we experimented with different ways for enhancing the dataset with KGEs and training the ML models, which are shown in details in the following section.

5. Proposed Approaches for Tabular Data Enrichment and ML Model Training

In this section, we outline the eight distinct approaches we explored for integrating KG embeddings into the training dataset, each designed to evaluate the impact of enriched semantic information on model performance.

5.1. Embeddings as ML Model Inputs (EmbedOnly)

We begin by our initial objective to investigate whether training a model on the vector representations generated from these KGs, using various embedding algorithms, could reveal underlying patterns and relationships within the data. Therefore, we define our first sub-hypothesis as follows.

> ***H1.1****: Using the embeddings alone, without any additional tabular data, could provide meaningful insights and capture latent relationships that enhance the model's predictive capability.*

To explore this, we first explored the EmbedOnly approach, focusing solely on the embeddings to assess their standalone effectiveness in capturing meaningful insights as shown in Figure 3.

For each instance p_i in the tabular data T, we have them represented as a subset $P \subseteq E$ of KG, where P represents the set of entities corresponding to instances in tabular data. The embedding function: $\phi : P \rightarrow \mathbb{R}^d$ is used to map each instance entity $p_i \in P$ to a d-dimensional vector space. Consequently, during both the training and testing phases only the embeddings $\{\phi(p) \mid p \in P\} \subset \mathbb{R}^d$ derived from the instance entities are used, as outlined in Algorithm 1. This ensures that the model is trained and evaluated only on the vector representations, capturing the semantic relationships within the KG relevant to the instances.

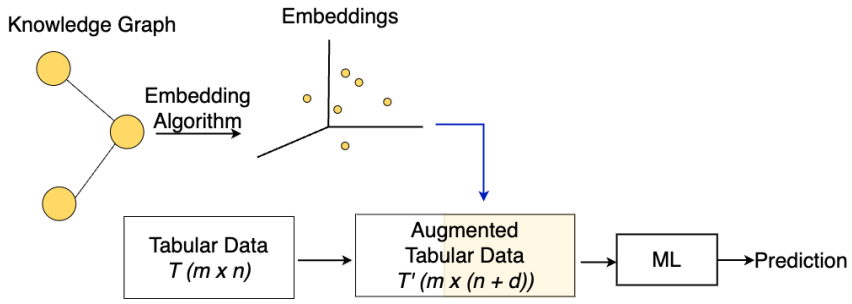

Fig. 4. Combining embeddings with tabular data as ML model inputs.

Algorithm 1 EmbedOnly

Require: Knowledge Graph $KG = (E, R)$, Tabular Data T, Embedding Function $\phi : E \cup R \rightarrow \mathbb{R}^d$, ML Model M
Ensure: Trained ML model using only embeddings for n splits

Initialize training data $X_{\text{train}} = []$, labels $Y_{\text{train}} = []$
Initialize test data $X_{\text{test}} = []$, labels $Y_{\text{test}} = []$
for each instance p_i in the training set T_{train} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Map entities to embeddings
    Append e_i to X_{train}
    Append label y_i corresponding to p_i to Y_{train}
end for
for each instance p_i in the test set T_{test} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Map entities (without y_i) to embeddings
    Append e_i to X_{test}
    Append label y_i corresponding to p_i to Y_{test}
end for
Train ML model M using X_{train} and Y_{train}
Evaluate M on X_{test} and Y_{test}
return Trained model M

5.2. *Combining Embeddings with Tabular Data Features (EmbedAugTab)*

Building upon EmbedOnly approach, we define our second sub-hypothesis as follows.

> ***H1.2**: Combining the KG-derived embeddings with traditional tabular data might enhance model performance by introducing additional relational information from the KG* structure.

This led us to design approaches that integrate both embeddings and tabular features, aiming to see if the KG information could complement and enrich the existing dataset. Thus, we investigated EmbedAugTab and other subsequent approaches that leverage embeddings for data augmentation based on this intuition.

EmbedAugTab approach involves training ML algorithms on datasets that integrate the original tabular data with additional columns derived from embeddings, as illustrated in Figure 4 and presented in Algorithm 2. For each instance p in the tabular dataset T, we augment T by appending the embedding vector $\phi(p)$, corresponding to the instance $p \in P$. The embedding vector $\phi(p)$ is generated using the embedding function $\phi : E \rightarrow \mathbb{R}^d$. This process yields an augmented tabular matrix T' with dimensions $m \times (n + d)$, where each row i contains the original features from T concatenated with the d-dimensional embedding vector $\phi(p)$. The resulting augmented matrix T' is then utilized to train ML models, leveraging both the original tabular features and the vector representations of the instances. In the healthcare domain, each instance p represents a patient, and the embedding vectors are added for

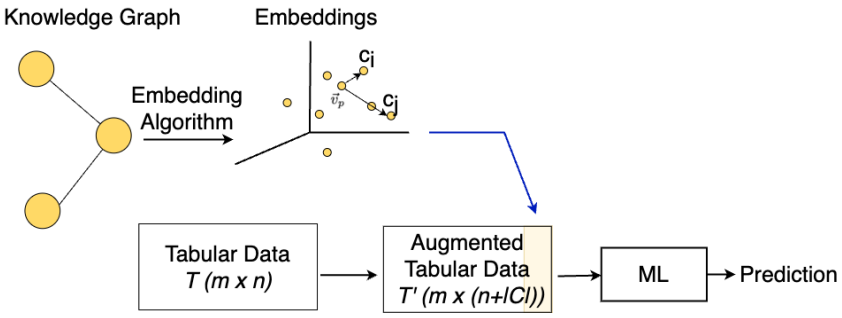

Fig. 5. Tabular dataset enrichment with distance measures from KG, used as ML model inputs.

each patient, in order to improve the models' ability to predict the presence or absence of specific diseases, such as heart disease or chronic kidney disease.

Algorithm 2 EmbedAugTab

Require: Knowledge Graph $KG = (E, R)$, Tabular Data T, Embedding Function $\phi : E \cup R \to \mathbb{R}^d$, ML Model M
Ensure: Trained ML model M for each T_{train}, T_{test} of n splits

Initialize training data $X_{\text{train}} = []$, labels $Y_{\text{train}} = []$
Initialize test data $X_{\text{test}} = []$, labels $Y_{\text{test}} = []$
for each instance p_i in T_{train} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    $X_{train} \leftarrow X_{train} \cup \text{Concatenate}(p_i, e_i)$ ▷ Append original features and embedding
end for
for each instance p_i in T_{test} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    $X_{test} \leftarrow X_{test} \cup \text{Concatenate}(p_i, e_i)$ ▷ Append original features and embedding
end for
Train ML model M using X_{train} and Y_{train}
Evaluate M on X_{test} and Y_{test}
return Trained model M

5.3. Tabular Dataset Enrichment with Distance Measures from Knowledge Graphs (DistAugTab)

Utilizing embedding vectors directly to augment the tabular data may introduce noise. Thus, our sub-hypothesis is defined as follows.

> ***H1.3**: Extracting specific structural information from the embedding space, such as distance matrices or cluster characteristics, might enhance model performance by providing more interpretable features for distance-based models.*

This led us to introduce the DistAugTab and ClustAugTab approachess, which aim to selectively extract meaningful information from the embeddings to improve the learning process.

In DistAugTab approach, we enhance the tabular dataset T by incorporating additional features derived from embedding-based distance calculations, as illustrated in Figure 5 and presented in Algorithm 3. For each instance p_i in the dataset T, we compute its embedding vector $\vec{v}_i$ using the embedding function ϕ. To further enrich the representation of each instance, we introduce $|C|$ additional columns, where C denotes the set of target classes.

The new columns are calculated by determining the Euclidean distance between the embedding vector $\vec{v}_i$ of instance p_i and the centroid $\vec{c}_{C_j}$ of each target class $C_j \in C$. These distance-based features are added to the augmented

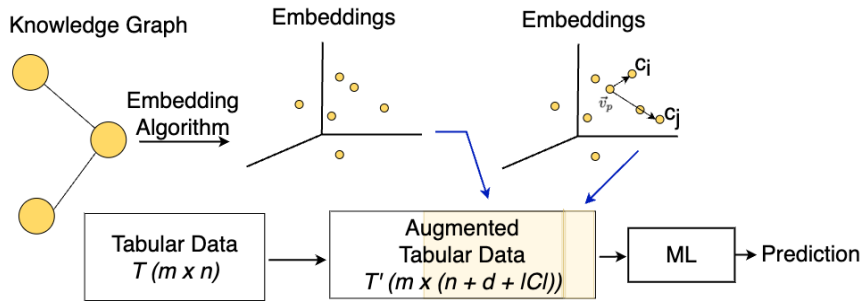

Fig. 6. Tabular dataset enrichment with distance measures from KG and vector embeddings, used as ML model inputs.

dataset T', resulting in an expanded dataset with dimensions $m \times (n + |C|)$, where m is the number of instances and n is the original number of features.

By including these distance features, we aim to capture how closely each instance's embedding aligns with the class centroids, thereby potentially improving the model's ability to differentiate between target classes. For example, in the healthcare domain, the target classes could represent the presence or absence of a disease $C = \{\text{disease}, \text{noDisease}\}$, where the distance features's aim is to help refine the model's predictions based on proximity to the centroids of the disease and no classes.

Algorithm 3 DistAugTab

Require: Knowledge Graph $KG = (E, R)$, Tabular Data T, Target Classes C, Embedding Function $\phi : E \cup R \to \mathbb{R}^d$, ML Model M
Ensure: Trained ML model M for each T_{test}, T_{test} of n splits
Initialize training data $X_{\text{train}} = []$, labels $Y_{\text{train}} = []$
Initialize test data $X_{\text{test}} = []$, labels $Y_{\text{test}} = []$
for each instance p_i in T_{train} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    for each class $C_j \in C$ **do**
        $d_{i,j} \leftarrow \|\vec{v}_i - \vec{c}_{C_j}\|_2$ ▷ Compute Euclidean distance between p_i and class centroid C_j
    end for
    $X_{\text{train}} \leftarrow X_{\text{train}} \cup \text{Concatenate}(p_i, d_{i,1}, \ldots, d_{i,|C|})$ ▷ Append original features and distances
end for
for each instance p_i in T_{test} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    for each class $C_j \in C$ **do**
        $d_{i,j} \leftarrow \|\vec{v}_i - \vec{c}_{C_j}\|_2$ ▷ Compute Euclidean distance between p_i and class centroid C_j
    end for
    $X_{\text{test}} \leftarrow X_{\text{test}} \cup \text{Concatenate}(p_i, d_{i,1}, \ldots, d_{i,|C|})$ ▷ Append original features and distances
end for
Train ML model M using X_{train} and Y_{train}
Evaluate M on X_{test} and Y_{test}
return Trained model M

5.4. Embedding and Distance Features Augmented Tabular Data (EmbedDistTabAug)

This approach augments the tabular dataset by incorporating both embedding vectors and distance-based features, as depicted in Figure 6 and presented in Algorithm 4. For each instance p_i, the augmented dataset T' is expanded by adding $d + |C|$ new columns, where d represents the embedding dimension and $|C|$ denotes the number of target classes. This results in an enhanced dataset with dimensions $m \times (n + d + |C|)$, combining the original features, embedding vectors, and distances to class centroids.

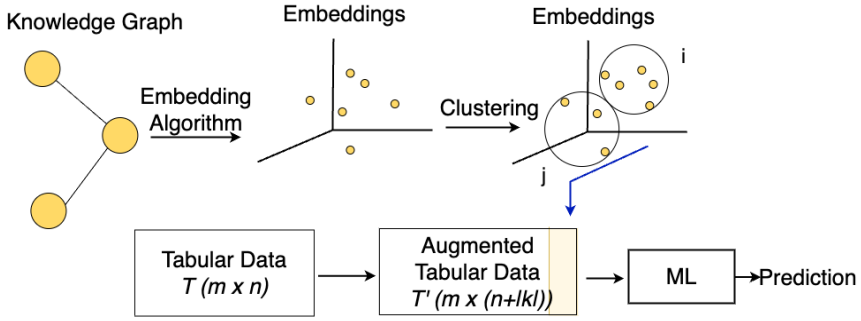

Fig. 7. Tabular dataset enrichment with embedding clusters' membership, used as ML model inputs.

Algorithm 4 EmbedDistTabAug

Require: Knowledge Graph $KG = (E, R)$, Tabular Data T, Target Classes C, Embedding Function $\phi : E \cup R \rightarrow \mathbb{R}^d$, ML Model M

Ensure: Trained ML model M for each T_{test}, T_{test} of n splits

Initialize training data $X_{\text{train}} = []$, labels $Y_{\text{train}} = []$
Initialize test data $X_{\text{test}} = []$, labels $Y_{\text{test}} = []$
for each instance p_i in T_{train} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    for each class $C_j \in C$ **do**
        $d_{i,j} \leftarrow \|\vec{v}_i - \vec{c}_{C_j}\|_2$ ▷ Compute Euclidean distance between p_i and class centroid C_j
    end for
    $X_{\text{train}} \leftarrow X_{\text{train}} \cup \text{Concatenate}(p_i, e_i, d_{i,1}, \ldots, d_{i,|C|})$ ▷ Append original features, embedding, and distances
end for
for each instance p_i in T_{test} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    for each class $C_j \in C$ **do**
        $d_{i,j} \leftarrow \|\vec{v}_i - \vec{c}_{C_j}\|_2$ ▷ Compute Euclidean distance between p_i and class centroid C_j
    end for
    $X_{\text{test}} \leftarrow X_{\text{test}} \cup \text{Concatenate}(p_i, e_i, d_{i,1}, \ldots, d_{i,|C|})$ ▷ Append original features, embedding, and distances
end for
Train ML model M using X_{train} and Y_{train}
Evaluate M on X_{test} and Y_{test}
return Trained model M

5.5. Tabular Dataset Enrichment with Embedding Clusters' membership (ClusterAugTab)

In this approach, referred to as EmbedClusterAugTab, we augment the tabular dataset by first computing embeddings for the data $E_{\text{train}} = \{\phi(p_i) | p_i \in T_{train}\}$ and then clustering these embeddings into n clusters using the K-means algorithm, as shown in Figure 7 and presented in Algorithm 6. Each instance $p_i \in T$ is assigned a cluster membership based on its embedding, which is added as an additional feature to the dataset. The augmented dataset T' now has dimensions $m \times (n+1)$, where the original n features are extended by one column representing the cluster membership derived from the embeddings. This enhanced dataset is then used to train the ML model, with the added cluster-level information facilitating the grouping of similar instances. By capturing these underlying patterns in the embeddings, the model can achieve improved predictive performance.

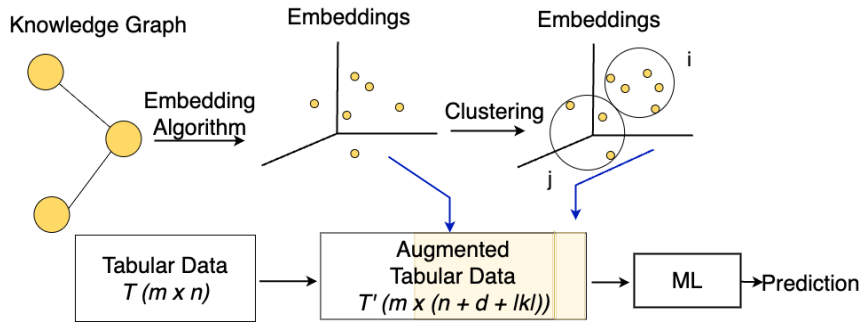

Fig. 8. Tabular dataset enrichment with embedding clusters' membership and vector embeddings, used as ML model inputs.

Algorithm 5 ClusterAugTab

Require: Tabular Data T, Number of Clusters n, K-means Clustering Algorithm, ML Model M
Ensure: Trained ML model M for each T_{test}, T_{test} of n splits

Initialize training data $X_{\text{train}} = []$, labels $Y_{\text{train}} = []$
Initialize test data $X_{\text{test}} = []$, labels $Y_{\text{test}} = []$
Compute embeddings for T_{train}: $E_{\text{train}} = \{\phi(p_i) | p_i \in T_{train}\}$
Initialize K-means with n clusters
Fit K-means on E_{train} to obtain cluster memberships
for each instance p_i in T_{train} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    $c_i \leftarrow$ K-means cluster for e_i ▷ Assign cluster membership based on embedding e_i
    $X_{\text{train}} \leftarrow X_{\text{train}} \cup \text{Concatenate}(p_i, c_i)$ ▷ Append original features and cluster membership
end for
for each instance p_i in T_{test} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    $c_i \leftarrow$ K-means cluster for e_i ▷ Assign cluster membership based on embedding e_i
    $X_{\text{test}} \leftarrow X_{\text{test}} \cup \text{Concatenate}(p_i, c_i)$ ▷ Append original features and cluster membership
end for
Train ML model M using X_{train} and Y_{train}
Evaluate M on X_{test} and Y_{test}
return Trained model M

5.6. Tabular Dataset Enrichment with Embeddings and Embedding Clusters' membership (EmbedClusterAugTab)

This approach, the tabular dataset is augmented by integrating both embedding vectors and cluster memberships, as shown in Figure 8 and detailed in Algorithm 6. For each instance p_i, the augmented dataset T' is expanded by appending both the d-dimensional embedding vector and the corresponding cluster membership, where d represents the embedding dimension. The resulting dataset has dimensions $m \times (n + d + 1)$, combining the original features, the learned embeddings, and the cluster assignments derived from the embeddings. This enriched representation enables the model to leverage both latent structure and group similarity for improved predictive performance.

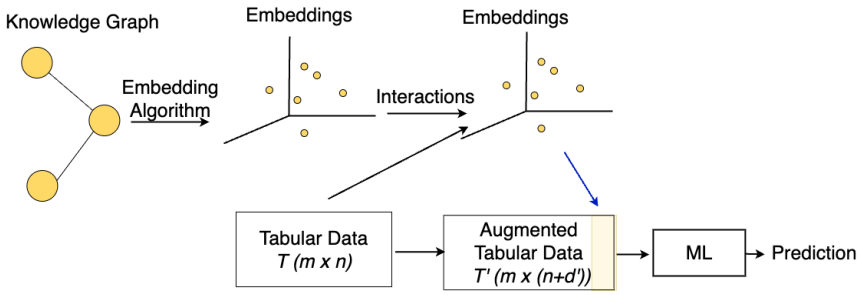

Fig. 9. Tabular dataset enrichment with feature interaction, used as ML model inputs.

Algorithm 6 EmbedClusterAugTab

Require: Tabular Data T, Number of Clusters n, K-means Clustering Algorithm, ML Model M
Ensure: Trained ML model M for each T_{test}, T_{test} of n splits

Initialize training data $X_{\text{train}} = []$, labels $Y_{\text{train}} = []$
Initialize test data $X_{\text{test}} = []$, labels $Y_{\text{test}} = []$
Compute embeddings for T_{train}: $E_{\text{train}} = \{\phi(p_i) | p_i \in T_{train}\}$
Initialize K-means with n clusters
Fit K-means on E_{train} to obtain cluster memberships
for each instance p_i in T_{train} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    $c_i \leftarrow$ K-means cluster for e_i ▷ Assign cluster membership based on embedding e_i
    $X_{\text{train}} \leftarrow X_{\text{train}} \cup \text{Concatenate}(p_i, e_i, c_i)$ ▷ Append original features, embeddings and cluster membership
end for
for each instance p_i in T_{test} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    $c_i \leftarrow$ K-means cluster for e_i ▷ Assign cluster membership based on embedding e_i
    $X_{\text{test}} \leftarrow X_{\text{test}} \cup \text{Concatenate}(p_i, e_i, c_i)$ ▷ Append original features, embeddings and cluster membership
end for
Train ML model M using X_{train} and Y_{train}
Evaluate M on X_{test} and Y_{test}
return Trained model M

5.7. Tabular Dataset Enrichment with Feature Interaction (InteraAugTab)

To further optimize the integration of KG information, we hypothesized that interactions between embeddings and existing features could reveal complex patterns. We define the sub-hypothesis as follows.

> ***H1.4**: Some classes may only be distinguishable through the combined effects of KG embeddings and tabular data.*

By developing approaches that compute these interaction terms, we aimed to enrich the feature space, enabling the model to capture dependencies arising from the integration of KG-derived and tabular features. This approach, implemented in the InteraAugTab approach, offers a multi-dimensional perspective that aims to improve accuracy and F2 score.

InteraAugTab approach augments the tabular dataset by incorporating interaction terms derived from the original features, as illustrated in Figure 9 and presented in Algorithm 7. For each instance p_i, the embedding vector e_i is computed using an embedding function ϕ. Interaction terms are then generated by element-wise multiplying each feature in p_i with each component of the embedding vector e_i. The augmented dataset T' thus contains the original

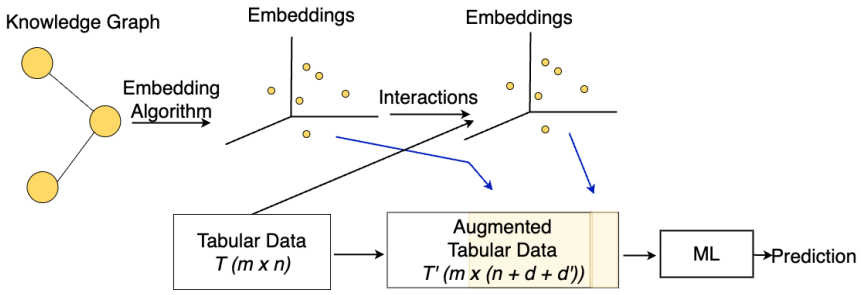

Fig. 10. Tabular dataset enrichment with feature interaction and vector embeddings, used as ML model inputs.

features and the interaction terms. This results in an enhanced dataset with dimensions $m \times (n + (n \times d))$, where n is the number of original features and d is the embedding dimension. The interaction terms enable the model to capture complex relationships between the original features and the latent information in the embeddings, potentially leading to improved predictive performance.

Algorithm 7 Feature Interaction Augmented Tabular Data (InteraAugTab)

Require: Tabular Data T, ML Model M
Ensure: Trained ML model M for each T_{test}, T_{test} of n splits
Initialize training data $X_{\text{train}} = []$, labels $Y_{\text{train}} = []$
Initialize test data $X_{\text{test}} = []$, labels $Y_{\text{test}} = []$
for each instance p_i in T_{train} **do**
    Generate interaction terms for p_i by computing pairwise products between selected feature pairs from p_i
    $X_{\text{train}} \leftarrow X_{\text{train}} \cup \text{Concatenate}(p_i, \text{interaction terms})$ ▷ Append original features and interaction terms
end for
for each instance p_i in T_{test} **do**
    Generate interaction terms for p_i by computing pairwise products between selected feature pairs from p_i
    $X_{\text{test}} \leftarrow X_{\text{test}} \cup \text{Concatenate}(p_i, \text{interaction terms})$ ▷ Append original features and interaction terms
end for
Train ML model M using X_{train} and Y_{train}
Evaluate M on X_{test} and Y_{test}
return Trained model M

5.8. Tabular Dataset Enrichment with Embedding and Feature Interaction (EmbedInteraAugTab)

In this approach, referred to as EmbedInteractionAugTab, we augment the tabular dataset by incorporating both the embedding vectors and the interaction terms between the original features and the embedding vectors, as shown in Figure 10 and presented in Algorithm 8. Similar to InteraAugTab approach, the embeddings are computed and the interaction terms. The augmented dataset T' thus contains the original features, the embedding vectors, and the interaction terms resulting in dimensions $m \times (n + d + (n \times d))$, where n is the number of original features and d is the embedding dimension.

Algorithm 8 EmbedInteractionAugTab

Require: Tabular Data T, Embedding Function $\phi : E \cup R \rightarrow \mathbb{R}^d$, ML Model M
Ensure: Trained ML model M for each T_{test}, T_{test} of n splits
Initialize training data $X_{\text{train}} = []$, labels $Y_{\text{train}} = []$
Initialize test data $X_{\text{test}} = []$, labels $Y_{\text{test}} = []$
for each instance p_i in T_{train} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    $X_{int} \leftarrow$ Compute interaction terms between original features and embedding e_i
    $X_{\text{train}} \leftarrow X_{\text{train}} \cup \text{Concatenate}(p_i, e_i, X_{int})$ ▷ Append original features, embedding, and interaction terms
end for
for each instance p_i in T_{test} **do**
    $e_i \leftarrow \phi(p_i)$ ▷ Compute embedding vector for instance p_i
    $X_{int} \leftarrow$ Compute interaction terms between original features and embedding e_i
    $X_{\text{test}} \leftarrow X_{\text{test}} \cup \text{Concatenate}(p_i, e_i, X_{int})$ ▷ Append original features, embedding, and interaction terms
end for
Train ML model M using X_{train} and Y_{train}
Evaluate M on X_{test} and Y_{test}
return Trained model M

To address the risk of high dimensionality, which can adversely affect the performance of certain models, we implemented a dimensionality reduction step using the PCA algorithm [1]. This reduction was specifically applied to approaches integrating embeddings, namely EmbedOnlyRed, EmbedAugTabRed, and EmbedDistAugTabRed. We define our sub-hypothesis as follows:

H1.5*: Reducing the dimensionality of the embedding-augmented datasets will improve model performance by eliminating redundant or noisy features, thereby retaining only the most informative ones.*

6. Experimental Analysis

In this section, we discuss the experimental goals that guide our investigation in Section 6.1 and in section 6.2 we discuss the experimental setup and materials used to achieve these goals.

6.1. Experimental Goals

The goal of our experimental evaluation is to investigate the use of KGs through knowledge graph embeddings to enhance the predictive performance of ML methods. We leverage the semantic structure of the ontologies, to represent the instances with more semantics and then through our proposed approaches use these to augment the tabular dataset for a better ML performance. The specific goals of our experiments are as follows:

Optimal Integration of KGs into ML Pipelines (RQ1): We examine effective methods for incorporating KGs into ML pipelines to improve model performance, with a particular emphasis on accuracy and F2 score. This entails analyzing the integration strategies that can enhance the predictive power of ML models.

Influence of KG Embedding Techniques (RQ2): We seek to understand how different KG embedding algorithms affect performance outcomes in ML models when utilized to enrich tabular data. This exploration focuses on identifying which embedding techniques yield the best enhancements in model accuracy and F2 score.

Comparative Analysis of ML Algorithms with KG-Enhanced Data (RQ3): We assess the relative performance of various ML algorithms when supplemented with KG-derived information. This analysis will highlight how distinct algorithms exploit KG semantics to boost the predictive performance.

Table 1
Details of the ontologies for heart and kidney disease domain.

Domain	Ontologies	Classes	Object prop.	Data prop.
Heart	Small	29	6	10
	Extended	1664	6	10
	Snomed	80	24	10
Kidney	Snomed	113	27	21

6.2. Experiment Setup

Datasets. In our experiments, we used two publicly available datasets from Kaggle: the *Heart Disease*[1] and *Chronic Kidney Disease*[2] datasets. Both datasets are used for binary classification tasks, where the goal is to predict the presence (*disease*) or absence (*no disease*) of the disease.

- Heart disease dataset consists of 303 instances, with 14 features capturing various patient health indicators relevant to diagnosing heart disease such as heart rate and cholesterol.
- Chronic kidney disease contains 400 instances and 25 features, capturing various health metrics related to chronic kidney disease such as blood pressure and albumin levels.

Both datasets contain a mix of categorical and numerical attributes, making them suitable for testing the integration of KGE with tabular data.

Ontologies. For the heart disease, we used three different ontologies:

- The *Small* ontology, denoted as $O = (C, R, H^C)$, is a handcrafted model derived from Trepan Reloaded [10] that encapsulates the features found in the Heart Disease dataset.
- The *Extended* ontology, represented as $O = (C', R', H^C)$, is an extention of an existing ontology [3] $O = (C, R, H^C)$ which incorporates additional features from the dataset.
- The *Snomed* ontology is derived as sub-ontology from the SNOMED-CT ontology [4]. This ontology was constructed using the methodology proposed by Chen et al. [7], which focuses on extracting relevant ontological structures from SNOMED-CT based on a predefined set of seed concepts required in the output. Initially, we selected the relevant concepts in the SNOMED-CT browser [5] that align with the dataset's features. These concepts served as seed concepts in the extraction process, ensuring the resulting ontology included them.

For chronic kidney disease, we only used on the third approach, extracting a sub-ontology from SNOMED-CT, due to the lack of ontologies specific to this domain. An overview of the ontologies used for both domains, including the count of classes and properties, is presented in Table 1.

KG embedding methods. We used four embedding methods: Node2Vec, RDF2vec, DistMult and TransH. The first two methods were selected as random-walk based models in the embedding landscape, while DistMult and TransH were chosen based on the findings in the Sem@K paper [23], which identified them as outperforming models from the semantic matching and geometric model families, respectively. Detailed descriptions of these models are provided in Section 2.

In Table 2, we illustrate the parameters used for the embedding methods, tailored to the specific characteristics of the KGs. We averaged performance across three embedding dimensions to provide a more robust evaluation of each method.

ML models. In our experiments, we used four models: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Extreme Gradient Boosting (XGB), and a simple feedforward Neural Network (NN). KNN and SVM were

[1] https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset
[2] https://www.kaggle.com/datasets/mansoordaku/ckdisease
[3] https://bioportal.bioontology.org/ontologies/HFO
[4] https://www.snomed.org
[5] https://termbrowser.nhs.ukmar

Table 2
Parameters for different KGE methods for different KGs.

Domain	KG		Node2Vec Param.			RDF2Vec Param.			TransH & DistMult
		dimens.	walk length	walks	window	depth	walks/node	window	params
Heart	Small	[64,128,100]	40	200	5	4	100	5	default
	Extended	[64,128,100]	60	200	10	6	150	10	default
	Snomed	[64,128,100]	50	200	7	5	100	7	default
Kidney	Snomed	[64,128,100]	50	200	7	10	100	7	default

Table 3
Parameter grid for ML methods.

Method	Parameter	(Grid) Values
KNN	n_neighbors	[20, 25, 30, 35, 40]
SVM	C; kernel; probability	[0.9, 1.0, 1.1, 1.2]; rbf; True
XGB	learning_rate	[0.08, 0.09, 0.1, 0.11]
NN	layers; activation; loss; optimizer	[32, 16, 1]; [relu, relu, sigmoid]; binary crossentropy; adam

chosen because they are distance-based, aligning with our hypothesis that KGEs, which are also distance-based, would enhance their performance. Whereas, XGB and NN were included to test the effect of KGEs on more complex, non-distance-based models.

To ensure robust evaluation, we used stratified 5-fold cross-validation, maintaining the same class distribution in each fold. For reproducibility, a fixed random seed was applied throughout the experiments. We initially experimented with a wide range of hyperparameters and, to reduce computational cost, we narrowed the range to focus on the best-performing configurations, as shown in Table 3. Results were averaged to ensure consistency across different configurations.

Evaluation metrics. In our experiments, we computed both accuracy and F2 score to assess model performance. We selected the F2 score as a key metric due to its relevance in disease prediction tasks, where maximizing true positive cases is critical for effectively identifying patients with the disease.

7. Results

In this section are shown the experiment results based on the experiment setup that we discussed in Section 6.2, starting with heart disease prediction, followed by kidney disease prediction. We show the concluding results for each research question.

7.1. Heart Disease Prediction

Table 4 shows the average accuracy and F2 scores for four different ML models (KNN, NN, SVM, XGB), comparing their baseline performance on tabular data alone with results from various methods of augmenting this data using embeddings generated by four KG embedding algorithms (Node2Vec, RDF2Vec, DistMult, TransH). In the following, the results are analyzed based on the research questions.

Investigating the impact of various methods for data augmentation through KGE The different methods of augmenting tabular data with KG embeddings yield mixed results across models. Approaches such as EmbedAugTab, DistAugTab and EmbedDistAugTab often provide the most performance improvements, especially for models such as XGBoost and NN. For example, DistAugTab when Node2Vec is being used to generate the embeddings significantly improved the F2 score of XGB from 75.19 (baseline) to 90.85, highlighting the ability of XGB to effectively use the additional distance features from KG embeddings.

Conversely, SVM and KNN tend to struggle with complex augmentation methods, showing lower gains and even losses in some cases, as they are less suited to high-dimensional data and the resulting feature complexity.

In the following we consider the effectiveness of different approaches based on the sub-hypotheses H1.1 to H1.5.

H1.1 Analysis: Our initial hypothesis (H1.1) proposed that using KG-derived embeddings alone (EmbedOnly) could provide meaningful insights by capturing latent relationships within the data. However, the results across all models and embedding algorithms contradict this hypothesis. The EmbedOnly approach consistently underperformed the baseline for each ML model, regardless of the KG embedding method used. For example, with Node2Vec, the F2 score of SVM dropped significantly from 77.18 (baseline) to 48.31, and similar trends were observed for other models and embeddings. Even when dimensionality reduction (EmbedOnlyRed) was applied to the embeddings, the results remained poor. This suggests that the standalone embeddings lack the richness of information provided by the original tabular features, which include more direct indicators of patient characteristics and clinical factors. Additionally, using embeddings alone may introduce complexity without clear connections to the target variable, making it difficult for the models to extract useful patterns.

H1.2 Analysis: our second hypothesis (H1.2) proposed that combining the KG embeddings with traditional tabular data (EmbedAugTab) would enhance predictive performance by adding relational information from the KG structure. This approach showed mixed results. In some cases, it led to modest improvements, such as with NN using RDF2Vec to generate the embeddings (F2 score improved from 77.44 to 78.64) or KNN with Node2Vec and DistMult. For SVM, there were F2 score gains when using any embedding algorithm and EmbedAugTabRed compared to the baseline, suggesting that the additional KG information could help refine decision boundaries for SVM's kernel-based approach. However, XGBoost often underperformed when embeddings were added (EmbedAugTab), with scores generally below the baseline. This could be due to XGBoost's preference for a simpler feature space where tabular data alone provides more direct information, making the additional, less structured KG-derived features more of a drawback than a help.

H1.3 Analysis: To address potential noise from directly using embeddings, H1.3 suggested that extracting specific structural features, such as distances from class centroids (DistAugTab) or clustering characteristics (ClustAugTab), would yield better results. The performance of DistAugTab, especially using Node2Vec to generate embeddings, supports this hypothesis, showing significant improvements over the baseline across NN, SVM, and XGBoost models. For instance, using Node2Vec to generate embeddings and then using DistAugTab approach for data augmentation boosted the F2 score of NN from 77.44% to 78.78% and XGBoost from 75.19% to 90.85%, indicating that distance-based features may help capture nuanced relationships between instances and classes that are relevant for classification. NN and SVM also performend well using DistMult and TransH for embedding generation and DistAugTab approach, likely because these models can benefit from the distance measures, making it easier to distinguish between similar instances.

The ClustAugTab approach, in the other hand also showed improvements, particularly with KNN. For example, using RDF2Vec with ClustAugTab led to better clustering of instances, resulting in improved accuracy for KNN and SVM from 81.02% to 81.18% and from 79.75% to 80.16%, respectively. KNN benefited from this approach as it relies on distance metrics to identify neighbors, and having meaningful clusters aligned with its decision-making process. Similarly, SVM showed better results using RDF2Vec for embedding generations and ClustAugTab approach for data augmentation, possibly because the cluster memberships served as a valuable feature that helped define clearer support vectors for class separation.

H1.4 Analysis: Sub-hypothesis (H1.4) suggests that certain classes in the data may be more effectively distinguished when interactions between KG embeddings and traditional tabular data are considered. This hypothesis assumes that there are complex dependencies between the relational information captured by the embeddings and the raw features of the tabular data. The results of the InteraAugTab and EmbedInteraAugTab approaches provide some support for this hypothesis. For example, SVM shows minor improvement in accuracy using interaction terms as additional features, from 79.75% (baseline) to 80.11%, 80.18% and 79.94% when Node2Vec, DistMult and TransH algorithms respectively are being used to generate the embeddings. Comparing between different embedding algorithms, from the table it is shown that RDF2Vec generated the most suitable embeddings to be used for IntraAugTab approach.

H1.5 Analysis: Sub-hypothesis (H1.5) suggests that reducing dimensionality would help models by removing irrelevant or noisy features, thus focusing learning on the most informative aspects of the data. The results show

mixed outcomes: while dimensionality reduction sometimes improved performance by simplifying the input space, it often failed to match the effectiveness of methods that used the full set of features without PCA.

For instance, while EmbedAugTabRed and EmbedDistAugTabRed helped reduce overfitting for XGBoost when using DistMult to generate the embeddings by eliminating redundant features, it did not outperform the EmbedAugTab and EmbedDistAugTab methods for KNN. This suggests that, while PCA can be useful for some models it might remove valuable information that more sophisticated models can use, highlighting a trade-off between feature simplification and richness.

Investigating the impact of embedding algorithm The choice of KG embedding algorithm has a significant impact on model performance across the various approaches. Each embedding method captures different aspects of the knowledge graph structure, influencing how well the derived embeddings integrate with the original tabular data and the model's ability to leverage this information.

From the results shown in Table 4 and the F2 score differences illustrated in Figure 11[6], it is evident that Node2Vec and RDF2Vec generally lead to more consistent performance improvements compared to DistMult and TransH, particularly when combined with approaches such as EmbedAugTab and DistAugTab. For example, Node2Vec embeddings with the EmbedDistAugTab approach provided the most notable gains across models, including SVM, and XGBoost. This improvement suggests that Node2Vec's random walk-based approach is effective in preserving local neighborhood information and graph structure, which seems to translate well into the feature space used by the models. The relational patterns it captures may align better with the tabular features, providing additional context that aids in classification.

RDF2Vec also showed good performance, particularly with EmbedAugTab and ClustAugTab approaches. Its ability to leverage RDF graph structures and preserve semantic relationships appears to be beneficial, especially for models such as NN and SVM.

In contrast, DistMult and TransH showed more variable results. While these methods performed well in specific scenarios—such as DistAugTab with DistMult or TransH, particularly for SVM and XGBoost—they were less consistent across different approaches. For example, while DistMult's tensor factorization approach allows it to capture specific types of relational patterns, this does not always translate into performance gains when used for approaches such as ClustAugTab, IntraAugTab or EmbedAugTab.

Moreover, the figures show that XGBoost's performance is particularly sensitive to the choice of embedding algorithm. While XGBoost generally excelled for approach DistAugTab or EmbedDistAugTab using Node2Vec, it underperformed with simpler methods such as EmbedAugTab when combined with TransH or DistMult. This suggests that XGBoost requires embeddings that add clear, structured relational information rather than purely dense vector representations. Thus, Node2Vec and RDF2Vec's ability to provide richer, more interpretable representations likely aligns better with XGBoost's learning mechanism.

In conclusion, the choice of embedding algorithm plays a crucial role in determining the success of different data augmentation approaches. RDF2Vec consistently provides more valuable representations for enhancing model performance across a range of methods, likely due to their strength in capturing both local and global graph structures. DistMult and TransH, while potentially effective in capturing specific relational patterns, exhibit more variability and require carefully chosen augmentation methods to translate their structural information into improved model performance. These findings emphasize that selecting the right embedding algorithm is critical, as it can significantly influence how well the additional relational data is integrated into the learning process.

Investigating the impact of KGs choice Figure 12 shows that the choice of ontology (Small, Extended, or Snomed) slightly affects model performance. Using Snomed ontology generally provides the highest accuracy and F2 scores, due to its clinically structured information from medical experts, highlighting its ability to enrich predictions. The Small KG yields the poorest results among the three ontologies, arguably due to its handcrafted nature by non-medical experts, which limits its depth and relevance to complex medical relationships.

[6]Note that EmbedOnly and EmbedOnlyRed are omitted from the figure, due to their consistently poor performance, which skewed the scale for the other approaches

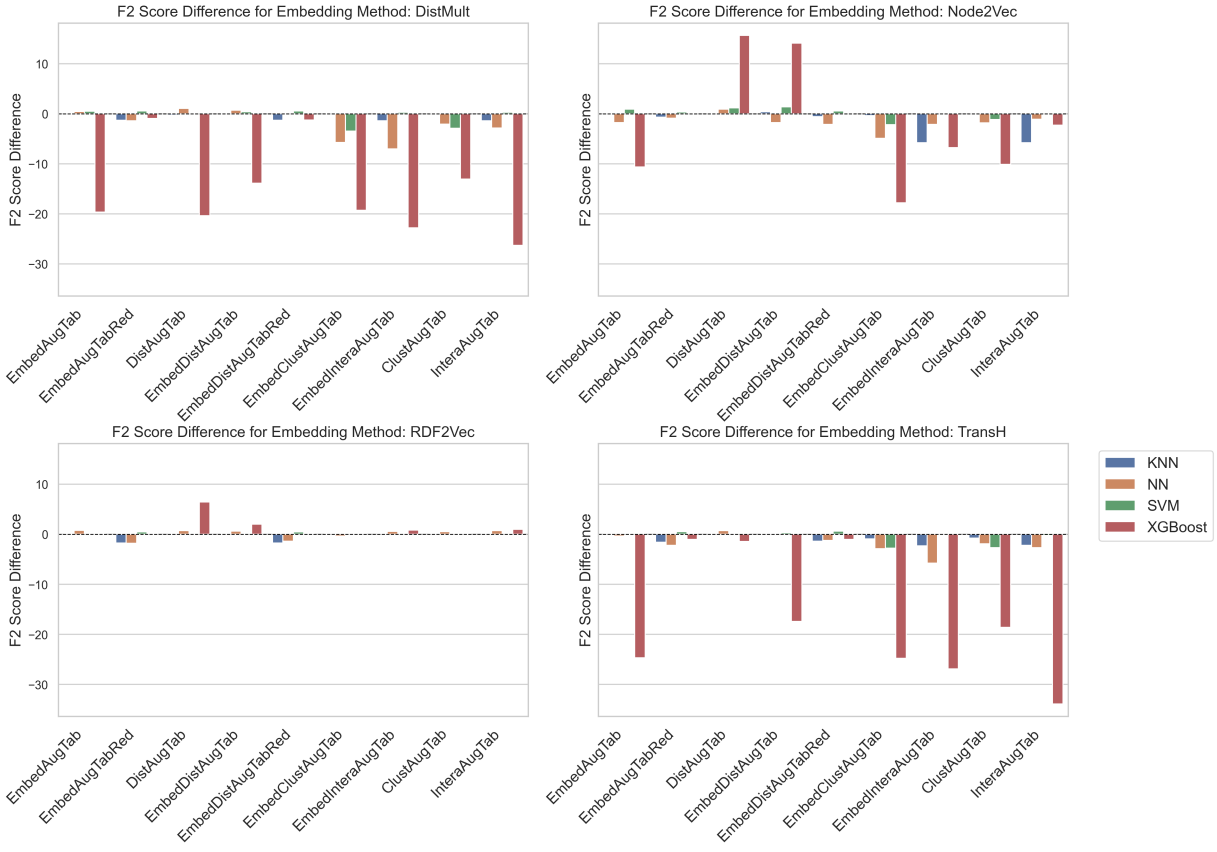

Fig. 11. F2 score differences relative to baseline across models and embedding methods, showing gain/loss for each approach for heart disease prediction

Investigating the performance of different ML models across various approaches and embedding algorithms Across the evaluated models, XGBoost and NN showed the most significant improvements when incorporating various KG augmentation methods. Specifically, XGBoost's performance saw the largest gains using the DistAugTab and EmbedDistAugTab approaches. For example, with Node2Vec embeddings combined using EmbedDistAugTab approach, XGBoost's F2 score increased from a baseline of 75.19% to 89.27%. This can be attributed to XGBoost's ability to effectively handle high-dimensional feature spaces, allowing it to extract valuable patterns from the distance-based features derived from the embeddings. However, XGBoost showed a lot of underperformance when the datasets were augmented with various approaches, especially when embeddings were generated with TransH and DistMult. This reduced performance may be due to the relational complexity in TransH and DistMult embeddings, which introduces interdependent features that XGBoost struggles to interpret independently.

On the other hand, KNN showed only slight performance gains when augmented with embeddings but maintained stable results across different approaches and embedding algorithms.

Looking at the average F2 scores in Table 5 when different embedding algorithms are used to generate the embeddings, we can observe that for four approaches SVM gained slightly better performance compared to the baseline, making it in general more suitable model that gains performance when additional data from KGs is being added, especially when computing the distances to the target classes, or when the vectors are added as such to augment the tabular data.

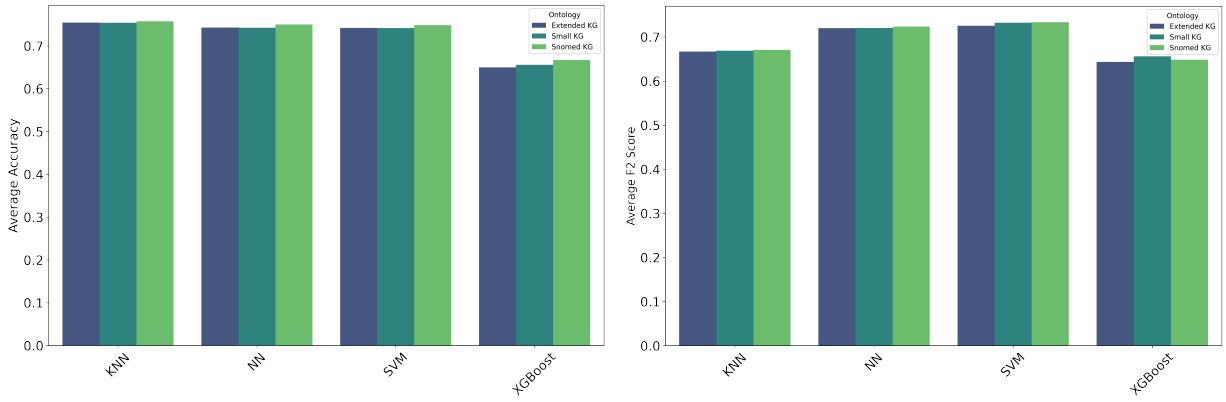

Fig. 12. Comparison of accuracy (left) and F2 (right) scores for different ML models across various KGs.

7.2. Kidney Disease Prediction

Table 6 shows the average accuracy and F2 scores fo different models, approaches and embedding methods in kidney disease prediction. In the following paragraphs, we will discuss the results based on the research questions, considering also the sub-hypothesis H1.1 - H1.5 from Section 5.

Investigating the impact of various methods for data augmentation through KGE From Table 6, we observe that adding distance-based features to tabular data improves ML model performance, especially for KNN and NN. Although the baseline is already high, enhancements such as distance-to-class, cluster membership features, and embedding vectors still boost performance. For example, KNN accuracy increases from 97% to 99.08% and 99.12% when the data is augmented with vector embeddings (EmbedAugTabRed) and with embeddings plus Euclidean distances to classes (EmbedDistAugTabRed), using TransH to generate embeddings. In the following we will discuss the hypothesis H1.1 - H1.5 based on the results.

H1.1 Analysis: As with heart disease prediction, using only embeddings (EmbedOnly) consistently underperforms compared to the baseline, regardless of the ML model or embedding method, contradicting the hypothesis. This suggests that embeddings alone provide less insight than tabular data for capturing relationships. Performance is particularly poor with RDF2vec embeddings, likely because RDF2vec focuses on structural patterns rather than the detailed, feature-specific information captured in tabular data.

H1.2 Analysis: In line with this hypothesis, the table shows that augmenting tabular data with embedding vectors (EmbedAugTab) generally results in similar or slightly improved accuracy and F2 scores compared to the baseline. For instance, for KNN, adding Node2Vec embeddings increases accuracy from 97% to 97.19% and the F2 score from 98.43% to 98.53%. Likewise, for NN, augmenting with RDF2Vec embeddings raises both accuracy and F2 scores from 99.92% and 99.96% to 100%.

H1.3 Analysis: This hypothesis suggests that adding structural features, such as distances to class centroids (DistAugTab) or clustering membership (ClustAugTab), also adding the embedding vectors (e.g., EmbedDistAugTab), should enhance performance. Structural features help capture relationships in the data by adding context about group distances, which is especially useful for proximity-based models such as KNN, SVM and NN. The results support this, showing performance generally remains consistent with or slightly better than the baseline. For example, for NN using Node2Vec embeddings, the DistAugTab approach increases accuracy and F2 score from 99.92% and 99.96% to 100%, as Node2Vec effectively captures neighborhood structures that align with these models' reliance on similarity. However, with TransH embeddings, which emphasize hierarchical relationships, ClustAugTab and EmbedClustAugTab slightly decrease accuracy and F2 scores for KNN, NN, and SVM, as these embeddings may introduce noise rather than meaningful distance-based information.

H1.4 Analysis: Similarly to the heart disease prediction results, Table 6 shows some results supporting the hypothesis that complex interactions between embeddings and raw tabular features can further improve model perfor-

mance. SVM generally maintained its 100% performance across different embedding algorithms. RDF2Vec showed to be the most compatible embedding algorithm to be used with our approaches, as it did not lead to performance drops with any approach. KNN on the other hand showed performance improvements when embeddings are used in those two approaches generated from DistMult, with accuracy improvement from 97.00% with only tabular data to 97.29% when interaction terms were included via the InterAugTab approach.

H1.5 Analysis: In the line with this hypothesis, suggesting that tabular dimensionality reduction can help eliminate noisy features, the results for kidney prediction show mixed outcomes. For KNN, applying the PCA algorithm consistently increases both accuracy and F2 scores compared to using the full feature set, particularly for the approaches EmbedDistAugTab and EmbedAugTabRed. This improvement is expected given KNN's struggles with high-dimensional data; it relies heavily on distance calculations, and PCA effectively enhances the quality of these calculations by reducing noise and emphasizing the most informative dimensions. For instance, when using embeddings generated with Node2Vec, dimensionality reduction in EmbedAugTab and EmbedDistAugTab boosts the accuracy from 97.19% and 97.21% to 99.10% in both cases.

Conversely, the performance of NN, SVM, and XGBoost generally worsened after the dimensionality reduction step. This could be attributed to the already high baseline accuracy (ranging from 99.75% to 100%); further reducing the dimensionality might eliminate features that, while not highly significant, still contribute to the model's performance. Additionally, these models can inherently manage high-dimensional data and may not benefit as much from PCA as KNN does. As a result, the reduced feature set may lack the nuanced information that these more complex models require for optimal performance.

Investigating the impact of embedding algorithm Figure 13 shows the F2 score differences relative to the baseline across models and embedding methods, highlighting gains and losses for each approach in kidney disease prediction. The approaches EmbedOnly, EmbedOnlyRed, and EmbedInteraAugTab, as well as InteraAugTab, were excluded from analysis due to their skewed performance, particularly when using DistMult and TransH to generate the embeddings, which demonstrated low performance.

From the figure we see that RDF2VEC was shown to be the best suited algorithm among the four embedding algorithms for our approaches. It generates effective embeddings particularly for KNN where the F2 score is increased for some of the approaches and stayed the same for the others. Moreover for the SVM model it maintained a perfect F2 score of 100%, in comparison to other embedding algorithms where the performance dropped. This could indicate that RDF2VEC caputes relevant features that enhance the SVM's ability to establish clear decision boundaries, ultimately resulting in higher predictive performance.

In contrast, Node2Vec generated effective embeddings for the KNN model, where it slightly improved performance. However, its utility decreased for SVM, XGBoost, and NN, often leading to slight performance drops. This suggests that while Node2Vec captures local structural information well, it may introduce noise or irrelevant features for models such as SVM and XGBoost.

Using DistMult to generate the embeddings achieved performance gains with KNN but resulted in decreased performance for SVM. This inconsistency indicates that while DistMult enhances KNN's ability to capture relationships, it introduces noise for SVM's decision-making process.

Similarly, TransH performed best with KNN, especially in the EmbedAugTabRed and EmbedDistAugTabRed approaches, yet showed weaker results for other models, particularly XGBoost. This discrepancy highlights that TransH may capture specific relational aspects beneficial for KNN but lacks the broader applicability needed for more complex models such as XGBoost.

Overall, our findings suggest that RDF2Vec algorithm is the optimal choice for embedding generation for augmenting data across various models due to its ability to enhance relevant feature representation. Conversely, Node2Vec is particularly advantageous for KNN, emphasizing the need to carefully select embedding algorithms based on the specific model and approach used to ensure the most effective performance enhancement.

Investigating the performance of different ML models across various approaches Table 7 presents the average F2 scores for various ML models in kidney disease prediction, showing the impact of different approaches for combining tabular data with embeddings, averaged across multiple embedding methods. KNN showed the most notable improvements when augmented with KG embeddings across different approaches, likely due to its weaker baseline performance compared to the rest and the suitability of distance-based metrics and dimensionality reduction for this

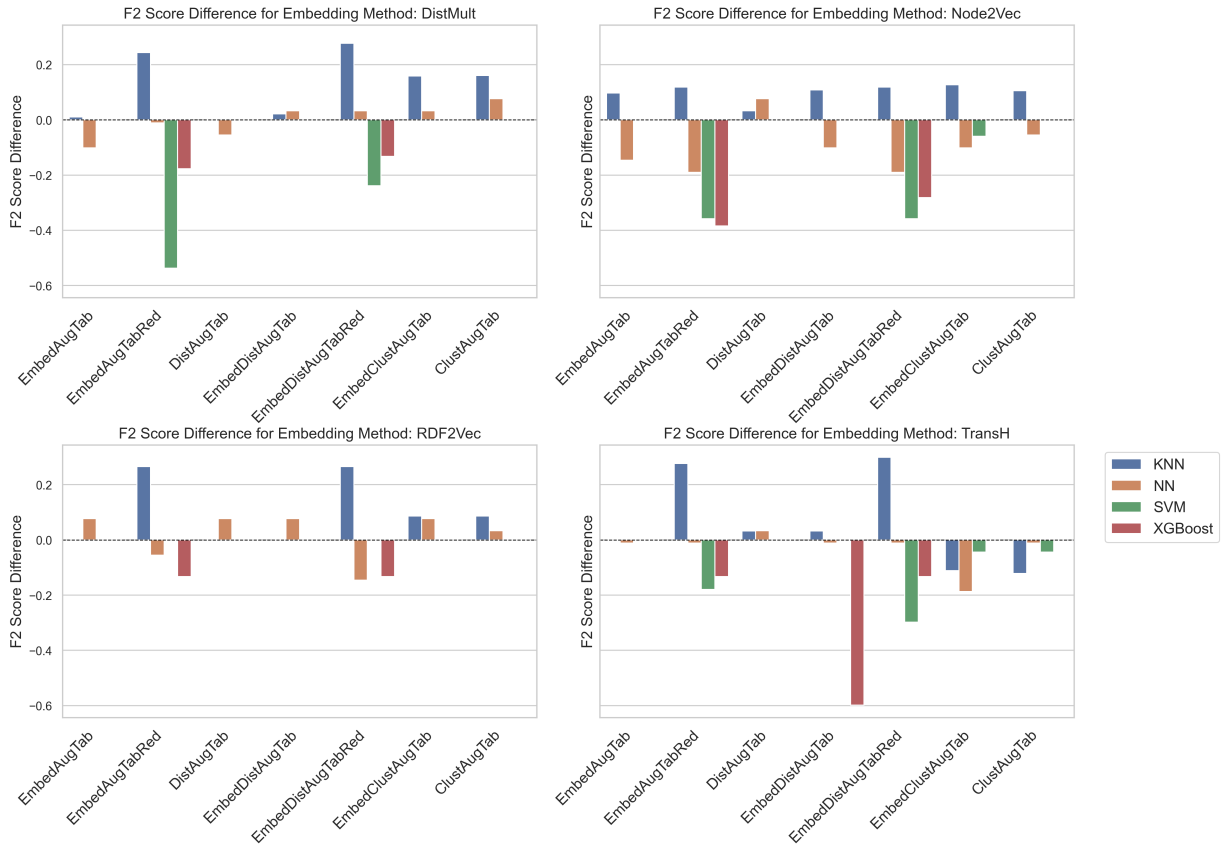

Fig. 13. F2 score differences relative to baseline across models and embedding methods, showing gain/loss for each approach for kidney disease prediction.

model. Excluding cases where only embeddings were used for training, NN generally maintained its performance with only slight drops in some approaches, suggesting that NN's ability to learn complex patterns is somewhat robust to variations in feature augmentation. SVM, which achieved a perfect F2 score (100%) with only tabular data, retained this performance in EmbedAugTab, DistAugTab, and EmbedDistAugTab. Similarly, XGBoost preserved its performance with the four embedding-augmented configurations, though it experienced slight declines in the remaining cases.

8. Conclusions and Future Work

In this paper, we proposed several innovative approaches to augment tabular data with semantic information by leveraging ontologies to capture domain semantics as shown in Figure 14. We utilized these ontologies to construct KGs, thereby enriching the datasets with structured ontological information. To make the knowledge graphs suitable for ML models, we employed knowledge graph embeddings to transform the graphs into a vector space representation. This process enhances the data used to train ML models by integrating domain-specific semantics, allowing the models to leverage contextual and relational information. Based on our experiment setup, we conducted experiments for heart and kidney disease prediction.

For RQ1, our experiments demonstrated that incorporating KG embeddings, particularly by augmenting tabular data with distance-based features to target classes, improves model performance in most of the cases. This enhancement is particularly evident in challenging domains such as chronic kidney disease, where accuracy and F2

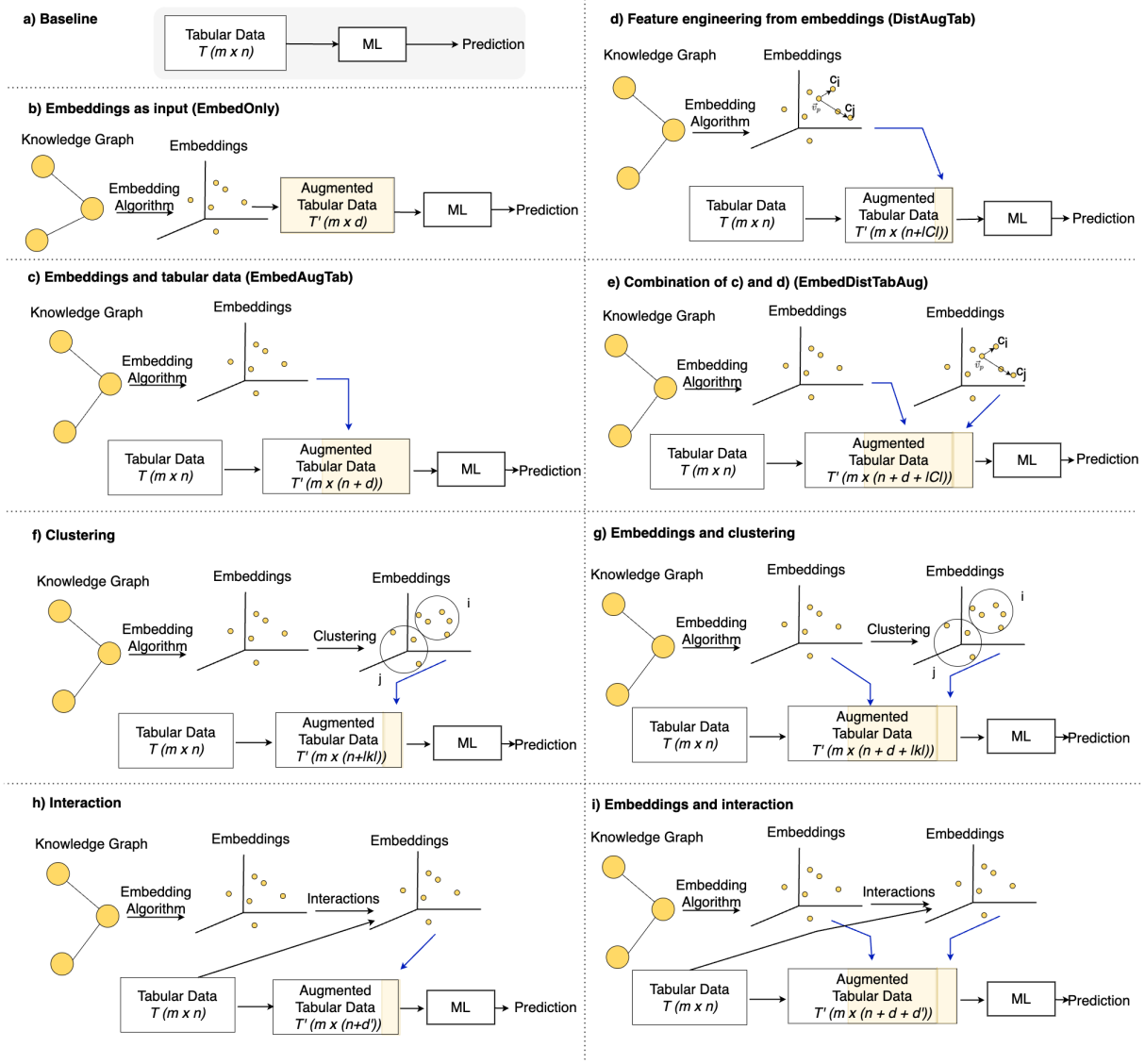

Fig. 14. Different ways of infusing the KG as input into ML pipeline.

scores improved despite limited room for improvement, underscoring the value of KG information for refining ML predictions, especially in data-sparse environments.

For RQ2, our findings indicate that RDF2Vec is the most effective embedding algorithm across models for both heart and kidney disease prediction, given its ability to capture relevant feature representations without performance drops. Node2Vec proved particularly beneficial for KNN in kidney disease prediction, while in heart disease prediction, Node2Vec enhanced XGBoost the most. However, XGBoost exhibited instability across approaches and embedding algorithms in both cases, suggesting the need for careful pairing of embedding methods and models.

For RQ3, in one hand for heart disease prediction overall on average SVM showed the most F2 score improvement across multiple approaches. Whereas on the other hand for kidney disease prediction, KNN showed the largest performance gains when enhanced with KG embeddings across various approaches, likely due to its weaker baseline performance and the suitability of distance-based metrics and dimensionality reduction, which complement KNN's neighbor-based approach.

Future work will explore the effectiveness of KGs across diverse domains, particularly those with limited data, by augmenting sparse datasets to address the data dependency issues in ML models. Additionally, we plan to assess the scalability of our methods based on data size and structure and experiment with more complex ML models to further optimize the integration of KG embeddings.

9. Acknowledgments

This work was supported by the FWF HOnEst project (V 745-N), FFG SENSE project (894802) and FAIR-AI project (904624).

References

[1] H. Abdi and L.J. Williams, Principal component analysis, *Wiley interdisciplinary reviews: computational statistics* **2**(4) (2010), 433–459.
[2] L. Ali, A. Niamat, J.A. Khan, N.A. Golilarz, X. Xingzhong, A. Noor, R. Nour and S.A.C. Bukhari, An optimized stacked support vector machines based expert system for the effective prediction of heart failure, *IEEE Access* **7** (2019), 54007–54014.
[3] K. Annervaz, S.B.R. Chowdhury and A. Dukkipati, Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing, *arXiv preprint arXiv:1802.05930* (2018).
[4] S. Bhatt, A. Sheth, V. Shalin and J. Zhao, Knowledge graph semantic enhancement of input data for improving AI, *IEEE Internet Computing* **24**(2) (2020), 66–72.
[5] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston and O. Yakhnenko, Translating embeddings for modeling multi-relational data, *Advances in neural information processing systems* **26** (2013).
[6] A. Breit, L. Waltersdorfer, F.J. Ekaputra, M. Sabou, A. Ekelhart, A. Iana, H. Paulheim, J. Portisch, A. Revenko, A.t. Teije et al., Combining machine learning and semantic web: A systematic mapping study, *ACM Computing Surveys* **55**(14s) (2023), 1–41.
[7] J. Chen, G. Alghamdi, R.A. Schmidt, D. Walther and Y. Gao, Ontology extraction for large ontologies via modularity and forgetting, in: *Proceedings of the 10th International Conference on Knowledge Capture*, 2019, pp. 45–52.
[8] P. Chittora, S. Chaurasia, P. Chakrabarti, G. Kumawat, T. Chakrabarti, Z. Leonowicz, M. Jasiński, Ł. Jasiński, R. Gono, E. Jasińska et al., Prediction of chronic kidney disease-a machine learning perspective, *IEEE access* **9** (2021), 17312–17334.
[9] C.G. Chute and C. Çelik, Overview of ICD-11 architecture and structure, *BMC medical informatics and decision making* **21**(6) (2021), 1–7.
[10] R. Confalonieri, T. Weyde, T.R. Besold and F.M. del Prado Martín, Using ontologies to enhance human understandability of global post-hoc explanations of black-box models, *Artificial Intelligence* **296** (2021), 103471.
[11] T. Dash, S. Chitlangia, A. Ahuja and A. Srinivasan, A review of some techniques for inclusion of domain-knowledge into deep neural networks, *Scientific Reports* **12**(1) (2022), 1040.
[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.
[13] S. El-Sappagh, F. Franda, F. Ali and K.-S. Kwak, SNOMED CT standard ontology based on the ontology for general medical science, *BMC medical informatics and decision making* **18** (2018), 1–19.
[14] A.d. Garcez and L.C. Lamb, Neurosymbolic AI: The 3 rd wave, *Artificial Intelligence Review* (2023), 1–20.
[15] M. Gaur, U. Kursuncu, A. Alambo, A. Sheth, R. Daniulaityte, K. Thirunarayan and J. Pathak, " Let me tell you about your mental health!" Contextualized classification of reddit posts to DSM-5 for web-based intervention, in: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 753–762.
[16] R. Gazzotti, C. Faron-Zucker, F. Gandon, V. Lacroix-Hugues and D. Darmon, Injecting domain knowledge in electronic medical records to improve hospitalization prediction, in: *The Semantic Web: 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2–6, 2019, Proceedings 16*, Springer, 2019, pp. 116–130.
[17] A. Grover and J. Leskovec, node2vec: Scalable feature learning for networks, in: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.
[18] T.R. Gruber, A translation approach to portable ontology specifications, *Knowledge acquisition* **5**(2) (1993), 199–220.
[19] A.P. Hassler, E. Menasalvas, F.J. García-García, L. Rodríguez-Mañas and A. Holzinger, Importance of medical data preprocessing in predictive modeling and risk factor discovery for the frailty syndrome, *BMC medical informatics and decision making* **19** (2019), 1–17.
[20] D. Herron, E. Jiménez-Ruiz and T. Weyde, On the Benefits of OWL-based Knowledge Graphs for Neural-Symbolic Systems, in: *Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning*, Vol. 3432, CEUR Workshop Proceedings, 2023, pp. 327–335.
[21] P. Hitzler, A. Eberhart, M. Ebrahimi, M.K. Sarker and L. Zhou, Neuro-symbolic approaches in artificial intelligence, *National Science Review* **9**(6) (2022), nwac035.
[22] Y.-X. Huang, Z. Sun, G. Li, X. Tian, W.-Z. Dai, W. Hu, Y. Jiang and Z.-H. Zhou, Enabling abductive learning to exploit knowledge graph, in: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023, pp. 3839–3847.

[23] N. Hubert, P. Monnin, A. Brun and D. Monticolo, Sem@*K*: Is my knowledge graph embedding model semantic-aware?, *Semantic Web* **14**(6) (2023), 1273–1309. doi:10.3233/SW-233508.

[24] M. Ivanović and Z. Budimac, An overview of ontologies and data resources in medical domains, *Expert Systems with Applications* **41**(11) (2014), 5158–5166.

[25] D. Jarrett, E. Stride, K. Vallis and M.J. Gooding, Applications and limitations of machine learning in radiation oncology, *The British journal of radiology* **92**(1100) (2019), 20190001.

[26] A. Jovic, M. Prcela and D. Gamberger, Ontologies in medical knowledge representation, in: *2007 29th International Conference on Information Technology Interfaces*, IEEE, 2007, pp. 535–540.

[27] R. Katarya and S.K. Meena, Machine learning techniques for heart disease prediction: a comparative study and analysis, *Health and Technology* **11** (2021), 87–97.

[28] H. Kautz, The third AI summer: Aaai robert s. engelmore memorial lecture, *AI Magazine* **43**(1) (2022), 105–125.

[29] J.D.M.-W.C. Kenton and L.K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: *Proceedings of naacL-HLT*, Vol. 1, Minneapolis, Minnesota, 2019, p. 2.

[30] U. Kursuncu, M. Gaur and A. Sheth, Knowledge infused learning (k-il): Towards deep incorporation of knowledge in deep learning, *arXiv preprint arXiv:1912.00512* (2019).

[31] Y. Lin, Z. Liu, M. Sun, Y. Liu and X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29, 2015.

[32] M. Llugiqi, F.J. Ekaputra and M. Sabou, Enhancing Machine Learning Predictions Through Knowledge Graph Embeddings, in: *International Conference on Neural-Symbolic Learning and Reasoning*, Springer, 2024, pp. 279–295.

[33] S. Mohan, C. Thirumalai and G. Srivastava, Effective heart disease prediction using hybrid machine learning techniques, *IEEE access* **7** (2019), 81542–81554.

[34] D. Moussallem, M. Arčan, A.-C.N. Ngomo and P. Buitelaar, Augmenting neural machine translation with knowledge graphs, *arXiv preprint arXiv:1902.08816* (2019).

[35] D.M. Pisanelli, *Ontologies in medicine*, Vol. 102, IOS press, 2004.

[36] K. Poulinakis, D. Drikakis, I.W. Kokkinakis and S.M. Spottswood, Machine-learning methods on noisy and sparse data, *Mathematics* **11**(1) (2023), 236.

[37] E.-H.A. Rady and A.S. Anwar, Prediction of kidney disease stages using data mining algorithms, *Informatics in Medicine Unlocked* **15** (2019), 100178.

[38] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu and M. Chen, Hierarchical text-conditional image generation with clip latents, *arXiv preprint arXiv:2204.06125* **1**(2) (2022), 3.

[39] P. Rani, R. Kumar, N.M.S. Ahmed and A. Jain, A decision support system for heart disease prediction based upon machine learning, *Journal of Reliable Intelligent Environments* **7**(3) (2021), 263–275.

[40] P. Ristoski and H. Paulheim, Rdf2vec: Rdf graph embeddings for data mining, in: *The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part I 15*, Springer, 2016, pp. 498–514.

[41] C. Ruiz, H. Ren, K. Huang and J. Leskovec, High dimensional, tabular deep learning with an auxiliary knowledge graph, *Advances in Neural Information Processing Systems* **36** (2024).

[42] M.K. Sarker, L. Zhou, A. Eberhart and P. Hitzler, Neuro-symbolic artificial intelligence, *AI Communications* **34**(3) (2021), 197–209.

[43] D. Shah, S. Patel and S.K. Bharti, Heart disease prediction using machine learning techniques, *SN Computer Science* **1** (2020), 1–6.

[44] A. Sheth, M. Gaur, U. Kursuncu and R. Wickramarachchi, Shades of knowledge-infused learning for enhancing deep learning, *IEEE Internet Computing* **23**(6) (2019), 54–63.

[45] A. Singhal et al., Introducing the knowledge graph: things, not strings, *Official google blog* **5**(16) (2012), 3.

[46] I. Szilagyi and P. Wira, An intelligent system for smart buildings using machine learning and semantic technologies: A hybrid data-knowledge approach, in: *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, IEEE, 2018, pp. 20–25.

[47] F. Van Harmelen and A. Ten Teije, A boxology of design patterns for hybrid learning and reasoning systems, *Journal of Web Engineering* **18**(1–3) (2019), 97–123.

[48] S. Vijayarani, S. Dhayanand and M. Phil, Kidney disease prediction using SVM and ANN algorithms, *International Journal of Computing and Business Research (IJCBR)* **6**(2) (2015), 1–12.

[49] M. Wang, L. Qiu and X. Wang, A survey on knowledge graph embeddings for link prediction, *Symmetry* **13**(3) (2021), 485.

[50] Z. Wang, J. Zhang, J. Feng and Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: *Proceedings of the AAAI conference on artificial intelligence*, Vol. 28, 2014.

[51] M. Wortsman, G. Ilharco, S.Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A.S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith et al., Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, in: *International conference on machine learning*, PMLR, 2022, pp. 23965–23998.

[52] A.L. Yadav, K. Soni and S. Khare, Heart Diseases Prediction using Machine Learning, in: *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, 2023, pp. 1–7.

[53] B. Yang, W.-t. Yih, X. He, J. Gao and L. Deng, Embedding entities and relations for learning and inference in knowledge bases, *arXiv preprint arXiv:1412.6575* (2014).

[54] P. Yildirim, Chronic Kidney Disease Prediction on Imbalanced Data by Multilayer Perceptron: Chronic Kidney Disease Prediction, in: *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2, 2017, pp. 193–198. doi:10.1109/COMPSAC.2017.84.

[55] C. Yin, R. Zhao, B. Qian, X. Lv and P. Zhang, Domain knowledge guided deep learning with electronic health records, in: *2019 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2019, pp. 738–747.

[56] K. Ziegler, O. Caelen, M. Garchery, M. Granitzer, L. He-Guelton, J. Jurgovsky, P.-E. Portier and S. Zwicklbauer, Injecting semantic background knowledge into neural networks using graph embeddings, in: *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, IEEE, 2017, pp. 200–205.

Table 4

Average accuracy and F2 scores, across different knowledge graphs, for various models, approaches, and embedding methods in heart disease prediction.

	KNN		NN		SVM		XGBoost	
Methods	Acc.	F2	Acc.	F2	Acc.	F2	Acc.	F2
Baseline	81.02	71.33	81.77	77.44	79.75	77.18	79.32	75.19
Node2Vec								
EmbedOnly	49.36	47.10	49.03	50.71	48.31	48.79	48.61	47.50
EmbedOnlyRed	49.36	47.10	49.00	50.48	48.19	48.52	48.93	50.90
EmbedAugTab	81.27	71.43	78.15	76.08	80.59	78.07	64.94	64.56
EmbedAugTabRed	80.60	70.63	81.02	76.93	79.34	77.53	79.72	75.24
DistAugTab	81.17	71.54	**82.17**	**78.78**	**81.81**	**78.36**	**92.51**	**90.85**
EmbedDistAugTab	**81.43**	**71.70**	77.71	76.10	**81.67**	**78.57**	**91.82**	**89.27**
EmbedDistAugTabRed	80.66	70.76	80.06	75.74	79.46	77.71	79.32	75.15
EmbedClustAugTab	81.17	70.97	72.43	72.96	76.10	75.01	55.32	57.39
EmbedInteraAugTab	79.07	65.56	75.95	75.70	80.12	76.95	69.22	68.40
ClustAugTab	81.21	71.16	78.01	76.03	77.58	76.02	62.93	65.05
InteraAugTab	79.06	65.55	78.81	76.77	80.11	77.00	74.18	72.90
RDF2Vec								
EmbedOnly	52.04	31.42	53.04	22.06	51.27	40.14	50.65	43.41
EmbedOnlyRed	52.04	31.42	53.34	21.84	51.27	40.16	51.00	43.18
EmbedAugTab	81.02	71.33	**82.07**	**78.64**	79.75	77.18	78.56	75.30
EmbedAugTabRed	79.95	69.59	80.32	76.06	79.32	77.63	78.77	75.25
DistAugTab	81.02	71.33	81.96	78.57	79.75	77.18	**84.38**	**81.62**
EmbedDistAugTab	81.02	71.33	81.85	78.46	79.75	77.18	80.60	77.20
EmbedDistAugTabRed	79.95	69.59	80.49	76.45	79.32	77.63	78.77	75.25
EmbedClustAugTab	**81.18**	71.12	81.44	77.48	**80.16**	**77.36**	78.64	75.34
EmbedInteraAugTab	81.02	71.33	81.81	78.43	79.76	77.18	79.33	76.03
ClustAugTab	**81.18**	71.12	81.81	78.38	**80.16**	**77.36**	79.10	75.22
InteraAugTab	81.02	71.33	**82.25**	**78.59**	79.75	77.18	79.23	76.15
DistMult								
EmbedOnly	48.04	62.88	46.43	64.43	47.14	68.57	47.78	54.97
EmbedOnlyRed	48.04	62.88	49.35	69.01	47.46	64.98	47.35	59.07
EmbedAugTab	81.07	**71.40**	80.35	78.30	80.03	**77.68**	49.60	55.53
EmbedAugTabRed	80.16	70.03	80.79	76.45	79.33	77.71	78.27	74.25
DistAugTab	80.88	71.04	**82.18**	**78.90**	**80.27**	77.39	53.42	54.84
EmbedDistAugTab	80.94	71.14	80.57	78.55	80.11	77.56	50.49	61.31
EmbedDistAugTabRed	80.16	70.02	81.30	77.69	79.34	77.71	78.16	73.92
EmbedClustAugTab	**81.39**	71.32	72.17	72.09	75.31	73.72	50.12	55.90
EmbedInteraAugTab	80.80	69.92	70.67	70.85	80.20	77.46	47.54	52.40
ClustAugTab	**81.43**	**71.45**	76.78	75.76	76.17	74.26	59.35	62.11
InteraAugTab	80.84	69.93	76.42	74.98	80.18	77.49	47.29	48.89
TransH								
EmbedOnly	48.22	53.12	47.88	59.80	48.55	59.32	47.57	50.25
EmbedOnlyRed	48.22	53.12	48.17	57.54	48.65	53.07	51.85	54.12
EmbedAugTab	**81.00**	71.24	80.68	77.51	79.89	77.32	48.59	50.51
EmbedAugTabRed	80.00	69.74	79.95	75.62	79.33	77.69	78.16	74.17
DistAugTab	80.98	71.27	**81.99**	**78.55**	**80.10**	77.30	75.34	73.77
EmbedDistAugTab	80.95	71.19	80.89	77.97	**80.11**	77.50	55.48	57.76
EmbedDistAugTabRed	80.08	69.95	80.72	76.61	79.38	**77.78**	78.20	74.17
EmbedClustAugTab	80.76	70.42	76.68	74.95	78.32	74.42	48.52	50.38
EmbedInteraAugTab	80.39	69.04	74.31	72.08	80.07	77.14	49.50	48.30
ClustAugTab	80.82	70.55	80.24	75.94	78.52	74.52	60.35	56.58
InteraAugTab	80.43	69.10	78.55	75.16	79.94	76.94	49.40	41.28

Table 5
Averages of F2 scores across embedding algorithms for different ML models and approaches in heart disease prediction

Model	KNN	NN	SVM	XGB
Baseline	71.33	77.44	77.18	75.19
EmbedOnly	48.63	48.84	54.21	49.03
EmbedOnlyRed	48.63	49.31	51.68	51.82
EmbedAugTab	**71.35**	77.22	**77.56**	61.47
EmbedAugTabRed	70.00	75.86	**77.64**	74.73
DistAugTab	71.30	**78.29**	**77.56**	**75.27**
EmbedDistAugTab	**71.34**	77.36	**77.70**	71.39
EmbedDistAugTabRed	70.08	76.21	**77.71**	74.62
EmbedClustAugTab	70.96	73.96	75.13	59.75
EmbedInteraAugTab	68.96	73.86	77.18	61.28
ClustAugTab	71.07	76.12	75.54	64.74
InteraAugTab	68.98	75.97	77.15	59.81

Table 6
Average accuracy and F2 scores for various models, approaches, and embedding methods in kidney disease prediction.

	KNN		NN		SVM		XGBoost	
Methods	Acc.	F2	Acc.	F2	Acc.	F2	Acc.	F2
Baseline	97.00	98.43	99.92	99.96	100.00	100.00	99.75	99.46
Node2Vec								
EmbedOnly	62.12	20.45	61.83	25.97	62.56	22.70	61.60	26.67
EmbedOnlyRed	62.12	20.45	61.58	24.62	62.92	23.26	61.42	25.43
EmbedAugTab	97.19	98.53	99.83	99.78	100.00	100.00	99.75	99.46
EmbedAugTabRed	**99.10**	**98.55**	99.75	99.73	99.83	99.64	99.28	99.08
DistAugTab	97.06	98.46	**100.00**	**100.00**	100.00	100.00	99.75	99.46
EmbedDistAugTab	97.21	98.54	99.92	99.82	100.00	100.00	99.75	99.46
EmbedDistAugTabRed	**99.10**	**98.55**	99.75	99.73	99.83	99.64	99.47	99.18
EmbedClustAugTab	97.38	**98.56**	99.92	99.82	99.97	99.94	99.75	99.46
EmbedInteraAugTab	96.54	98.03	98.83	98.44	99.06	97.96	99.75	99.46
ClustAugTab	97.27	98.54	99.75	99.87	100.00	100.00	99.75	99.46
InteraAugTab	96.46	97.99	99.00	98.39	99.14	98.14	99.75	99.46
RDF2Vec								
EmbedOnly	59.45	8.59	62.50	4.10	56.64	13.90	53.19	26.59
EmbedOnlyRed	59.45	8.59	56.25	7.75	56.58	13.89	53.83	27.52
EmbedAugTab	97.00	98.43	**100.00**	**100.00**	100.00	100.00	99.75	99.46
EmbedAugTabRed	**99.06**	**98.70**	99.75	99.87	100.00	100.00	99.50	99.33
DistAugTab	97.00	98.43	**100.00**	**100.00**	100.00	100.00	99.75	99.46
EmbedDistAugTab	97.00	98.43	**100.00**	**100.00**	100.00	100.00	99.75	99.46
EmbedDistAugTabRed	**99.06**	**98.70**	99.83	99.78	100.00	100.00	99.50	99.33
EmbedClustAugTab	97.17	98.52	**100.00**	**100.00**	100.00	100.00	99.75	99.46
EmbedInteraAugTab	97.00	98.43	**100.00**	**100.00**	100.00	100.00	99.75	99.46
ClustAugTab	97.17	98.52	99.92	99.96	100.00	100.00	99.75	99.46
InteraAugTab	97.00	98.43	**100.00**	**100.00**	100.00	100.00	99.75	99.46
DistMult								
EmbedOnly	66.09	46.62	59.17	44.73	64.17	16.95	55.83	26.58
EmbedOnlyRed	66.09	46.62	51.88	74.79	58.75	11.63	63.75	4.13
EmbedAugTab	97.02	98.44	99.92	99.82	100.00	100.00	99.75	99.46
EmbedAugTabRed	**99.08**	**98.68**	99.83	99.91	99.75	99.46	99.42	99.29
DistAugTab	97.00	98.43	99.75	99.87	100.00	100.00	99.75	99.46
EmbedDistAugTab	97.04	98.45	99.92	99.96	100.00	100.00	99.75	99.46
EmbedDistAugTabRed	**99.08**	**98.71**	99.92	99.96	99.89	99.76	99.50	99.33
EmbedClustAugTab	97.44	98.59	99.92	99.96	100.00	100.00	99.75	99.46
EmbedInteraAugTab	97.27	98.27	98.25	96.12	99.72	99.40	95.84	89.86
ClustAugTab	97.38	98.59	**100.00**	**100.00**	100.00	100.00	99.75	99.46
InteraAugTab	97.29	98.35	98.00	95.53	99.86	99.70	98.33	96.29
TransH								
EmbedOnly	41.82	67.08	45.81	58.45	56.64	33.89	44.73	63.87
EmbedOnlyRed	41.82	67.08	39.79	62.73	67.50	85.23	45.03	62.13
EmbedAugTab	97.00	98.43	99.83	99.91	100.00	100.00	99.75	99.46
EmbedAugTabRed	**99.08**	**98.71**	99.83	99.91	99.92	99.82	99.50	99.33
DistAugTab	97.06	98.46	99.92	99.96	100.00	100.00	99.75	99.46
EmbedDistAugTab	97.06	98.46	99.83	99.91	100.00	100.00	98.78	98.86
EmbedDistAugTabRed	**99.12**	**98.73**	99.83	99.91	99.86	99.70	99.50	99.33
EmbedClustAugTab	96.92	98.32	99.50	99.74	99.92	99.96	99.75	99.46
EmbedInteraAugTab	96.94	98.36	99.25	98.64	99.83	99.64	86.85	66.82
ClustAugTab	96.90	98.31	99.83	99.91	99.92	99.96	99.75	99.46
InteraAugTab	97.00	98.40	99.00	98.11	100.00	100.00	87.93	69.47

Table 7
Averages of F2 scores across embedding algorithms for different ML models and approaches in kidney disease prediction

Model	KNN	NN	SVM	XGBoost
Baseline	98.43	99.96	100	99.46
EmbedOnly	35.69	33.31	21.86	35.93
EmbedOnlyRed	35.69	42.47	33.50	29.80
EmbedAugTab	**98.46**	99.88	**100.00**	**99.46**
EmbedAugTabRed	**98.66**	99.86	99.73	99.26
DistAugTab	**98.45**	**99.96**	**100.00**	**99.46**
EmbedDistAugTab	**98.47**	99.92	**100.00**	99.31
EmbedDistAugTabRed	**98.67**	99.84	99.78	99.29
EmbedClustAugTab	**98.50**	99.88	99.97	**99.46**
EmbedInteraAugTab	98.27	98.30	99.25	88.90
ClustAugTab	**98.49**	99.93	99.99	**99.46**
InteraAugTab	98.29	98.01	99.46	91.17