Graph Machine Learning for Fast Product Development from Formulation Trials

Manuel Dileo[1(✉)], Raffaele Olmeda[2], Margherita Pindaro[2], and Matteo Zignani[1]

[1] Department of Computer Science, University of Milan, Milan, Italy
{manuel.dileo,matteo.zignani}@unimi.it
[2] Intellico s.r.l, Milan, Italy
{raffaele.olmeda,margherita.pindaro}@intellico.ai

Abstract. Product development is the process of creating and bringing a new or improved product to market. Formulation trials constitute a crucial stage in product development, often involving the exploration of numerous variables and product properties. Traditional methods of formulation trials involve time-consuming experimentation, trial and error, and iterative processes. In recent years, machine learning (ML) has emerged as a promising avenue to streamline this complex journey by enhancing efficiency, innovation, and customization. One of the paramount challenges in ML for product development is the models' lack of interpretability and explainability. This challenge poses significant limitations in gaining user trust, meeting regulatory requirements, and understanding the rationale behind ML-driven decisions. Moreover, formulation trials involve the exploration of relationships and similarities among previous preparations; however, data related to formulation are typically stored in tables and not in a network-like manner. To cope with the above challenges, we propose a general methodology for fast product development leveraging graph ML models, explainability techniques, and powerful data visualization tools. Starting from tabular formulation trials, our model simultaneously learns a latent graph between items and a downstream task, i.e. predicting consumer-appealing properties of a formulation. Subsequently, explainability techniques based on graphs, perturbation, and sensitivity analysis effectively support the R&D department in identifying new recipes for reaching a desired property. We evaluate our model on two datasets derived from a case study based on food design plus a standard benchmark from the healthcare domain. Results show the effectiveness of our model in predicting the outcome of new formulations. Thanks to our solution, the company has drastically reduced the labor-intensive experiments in real laboratories and the waste of materials.

Keywords: Product Development · Structure Learning · XAI for tabular data

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-70378-2_19.

A. Bifet et al. (Eds.): ECML PKDD 2024, LNAI 14949, pp. 303–318, 2024.
https://doi.org/10.1007/978-3-031-70378-2_19

1 Introduction

Product development refers to the systematic process of designing, creating, and introducing new or improved products into the market. A fundamental step of this process is represented by the formulation trials, in which the research and development (R&D) department of industrial companies experiments with various ingredients, proportions, physical properties, and other factors to determine the optimal combination that meets the desired specifications and performance criteria. Conventional approaches to formulation trials typically utilize labor-intensive experimentation, trial and error, and iterative procedures, which can take several weeks to meet a desired formulation.

Over the past few years, machine learning (ML) has emerged as a promising solution for simplifying this process and enhancing efficiency, innovation, and customization. A main challenge in ML for product development is the models' lack of interpretability and explainability. This limitation poses significant burdens in gaining user trust, fulfilling regulatory standards, and understanding the logic behind ML-driven decisions. Moreover, formulation trials involve the exploration of relationships and similarities among previous preparations or solutions; but, data related to these formulations are typically stored in tables without explicit relationships between trials.

To cope with the above challenges, we propose a general methodology for fast product development leveraging graph machine learning models, explainability techniques, and powerful data visualization tools. Starting from tabular formulation trials, our model simultaneously learns a latent graph between items and a downstream task, i.e. predicting consumer-appealing properties of a formulation. This choice enables the model to explore latent correlations among formulations and feature values and allows transferable knowledge between similar product lines or iterative design processes. As a further step, explainability techniques based on graphs, perturbation, and sensitivity analysis effectively support the R&D department in their formulation process. Specifically, global-level explanations - related to the overall predictions of the model - allow them to identify the most influential characteristics for obtaining a certain property, while graph and single-level explanations - related to the prediction of single formulations - effectively support the users in identifying new recipes, the impact of changes in existing recipes, and the restocking of ingredients. The current solution is in the deployment phase and is offered to industrial companies through customized web applications.

As case study, we present the application of the above solution in the context of food design; presenting two real-world datasets collected in collaboration with a renowned company in the sector. In this scenario, industrial researchers are interested in finding recipes that satisfy specific sensorial properties of the product over time, given the composition of its ingredients and its physical properties. We model the problem of predicting sensorial properties as a multi-regression task. We evaluate our model on the two datasets, exhibiting the effectiveness of our solution compared to common tabular, graph, and structure learning approaches. Moreover, an ablation study highlights that the graph machine

learning approach plays a crucial role in obtaining the best performance against the baselines. Finally, we also evaluate our method on a standard benchmark dataset for structure learning in the healthcare domain, showing an increase in performance compared to state-of-the-art models.

From the client side, our solutions are positively impacting different business and production metrics: the company is expected to reduce the labor-intensive experiments from 150 tests before approval to 30, the waste of materials dropped up to 30%, and the time to market has passed from seven to two months. We believe our work highlights a promising avenue for graph machine learning on general formulation trials in product development.

2 Background

In this section, we briefly provide background on graph neural networks. Then, we describe related works on the use of graph neural networks for product development and graph structure learning for tabular data, highlighting some works that provide also explainability techniques for the task.

Graph Neural Networks. Graph Neural Networks (GNNs) [18] are neural networks specifically designed to handle graph-structured data. Thanks to their ability to propagate and aggregate information across nodes and edges within a graph, they excel at capturing relationships and dependencies in graph-structured information to generate a vector-based representation for nodes, leveraging also node and/or edge-level attributes. This versatility has led to their application in diverse domains such as social network analysis [3], recommendation systems [7], or bioinformatics [5]. Modern GNNs rely on the 1-hop message-passing framework [6] for processing graph data. Specifically, given a graph $\mathcal{G} = (V, E)$, the representation of a node $v \in V$ at the l-layer of a GNN is obtained as a combination of the representation of the node v at layer $l-1$ and aggregation of the representation of the nodes in the 1-hop neighborhood of v, where layer zero is represented by the initial node features. Different kinds of aggregation and combination functions can be considered to build different GNN layers.

GNNs for Product Development. Recently, Graph Neural Networks have been successfully applied in product development. In particular, GNNs are used for product design of items that can be naturally represented as graphs, meaningfully processed as 3D data, or whose interactions with other elements are explicitly defined. For instance, Bian *et al.* [1] formulate the material selection problem as a binary node-level classification task over the assembly of Computer-aided design (CAD) projects, modeled as a graph of material components, and leverage GNNs to obtain representation for materials. CAD models can be also represented as 3D structures, e.g. using Point Clouds [12], whose representation can be learned using GNNs [22]. In the context of drug design, the interaction

between co-prescribed drugs and drug-target protein can be leveraged by GNNs for drug repurposing and identification of side effects [5]. In the last few years, GNNs were also employed for scaling deep learning for materials discovery and improving the modeling of material properties given their molecular structure or composition [17]. Although all these methods have shown successful applications of GNNs in product development, most of them assume a fixed graph structure between elements or explicit interactions between products. But, in most cases, formulation trials in product development are typically stored as tables and their complex relationships (e.g. recipe similarities, sharing knowledge between experiments) are not explicitly modeled. Moreover, in the context of food design, where industrial researchers are interested in consumer-appealing taste properties, the use of 3D data structures would not be as meaningful as in the context of manufacturer design.

Graph Learning for Tabular Data. In recent years, the community has underlined a critical gap in deep learning for tabular data: the lack of representation of latent correlations among data instances and feature values [14]. GNNs, due to their ability to model relationships between diverse elements of tabular data, have attracted considerable interest and have been applied across various steps of tabular data processing [14]. In particular, graph structure learning methods [24] aim to jointly learn an optimized latent graph structure among elements and an element-level downstream task. Among these approaches, only a few provide also explanations for the obtained results. For instance, Verdone *et al.* [21] utilizes GNNExplainer [23] for providing explainable spatio-temporal predictions for multi-site photovoltaic energy production. In the healthcare domain, Kazi *et al.* [9] provides an attention-based mechanism for interpretable Alzhaimer's disease predictions, while Li *et al.* [15] utilizes interpretable feature learning for Parkinson's disease diagnosis. To the best of our knowledge, no explainable graph machine-learning techniques have been applied in the context of product development and food design.

3 Dataset and Case-Study

We present a case study in the context of product development for the food industry introducing two real-world datasets collected in collaboration with a renowned company in the sector, Perfetti Van Melle (PVM). In particular, the PVM Lainate Labs is the entity in charge of running the analysis and trials necessary to create new recipes and formulations for their products. Given formulations of previously tested products and their corresponding consumer-appealing properties, the primary goal of our solution is to speed up product development by predicting the property values of new formulations. This allows R&D departments and labs to fast discover new potential formulations for a desired property. The next subsections will detail the case study and datasets used to test our solution.

Case-Study. Product development in the food industry involves traditionally several steps, which can be summarized as follows: (i) *Product Design*: defining in detail the consumer-appealing characteristics of the new product, based on several factors such as customer feedback, new ingredients availability, new strategies, and other market needs; (ii) *Recipe formulation*: creating the recipe to meet the desired design. This step is based mostly on experience matured over the years by the members of the research and development (R&D) department; (iii) *Laboratory Analysis*: measuring the physical properties of the product utilizing highly specialized technical equipment (e.g. rheometers) and monitoring the various aspects of the product through dashboards; (iv) *Process iteration*: at the end of the described steps, the overall quality of the product is evaluated. If the metrics are not aligned with the desired properties, the whole process is repeated iteratively. This process can take several weeks to obtain a desired recipe formulation. The goal of our solution is to allow the company to abandon an iteration-based process in favor of a data-driven approach, in which AI techniques can suggest a new product design, given the past trials and the desired consumer-appealing characteristics. Specifically, a machine learning model is trained to predict the properties of new formulations. Afterwards, explainability and advanced data analysis techniques are leveraged to support the R&D department in designing new recipes.

Dataset. We collected two datasets derived from the formulation trials tested by the R&D department of PVM. Each formulation trial is described by the following groups of features: (i) *Raw materials*: the ingredients of the recipe. They are values between zero and one, each value represents the percentage of a certain ingredient in the recipe. As a consequence, the sum of these features for each row must be one. Some recipes require specific ingredients, while others may not utilize all of them; (ii) *Physical properties*: features derived from the laboratory analysis (e.g. using rheometers). An example of physical property is the viscosity of the product; (iii) *Sensorial properties*: the consumer-appealing properties to predict. They are obtained thanks to a panel of several people who taste the recipe and measure a particular sensation multiple times over a defined time interval. In this work, we focus on trials for "malleability" and "toughness" sensorial properties. Hence, overall, we obtained two datasets, each of them consisting of one hundred formulation trials described by forty raw materials and twenty-three physical properties. More information about the dataset and its preprocessing can be found in the supplementary material.

4 Methodology

In this section, we describe the proposed methodology for predicting the outcome of new formulation trials for specific desired product properties, which allows fast product development for industrial labs. The methodology leverages the characteristics of previously tested formulations and an inferred underlying graph structure that captures the similarities between previous laboratory trials.

Figure 1 shows the pipeline of our solution. Starting from the characteristics of previously tested products, modeled as tabular data, the *Differentiable Graph Module* (DGM)-based model [8] simultaneously learns, in an end-to-end fashion, the downstream task, i.e. the properties of a product, and an optimal underlying latent graph. The *Explainer* shows the most important product characteristics for a certain expected property and the most important nodes, i.e. previously tested recipes, for reaching the desired characteristic. The *Explorer* allows the R&D department to visualize the obtained graph for investigating new possible trials. Finally, the *Simulation module* leverages the DGM module at inference time to simulate the outcome of new formulation trials before getting in real laboratories.

The current solution is in the deployment phase and is offered to industrial companies through customized web applications. The next subsections describe the pipeline modules and the main features of the web application in detail.

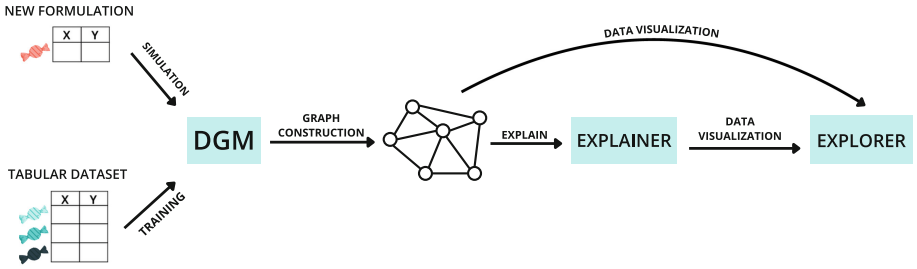


Fig. 1. Pipeline of our methodology for explainable prediction outcome of formulation trials.

4.1 DGM-Based Model

We model formulation trials as a $\mathbf{X} \in \mathbb{R}^{N \times d}$ matrix, where N is the number of formulations and d is the number of features (raw materials and physical properties). Each row of $\mathbf{X}$ has an associated vector $\mathbf{y} \in \mathbb{R}^z$ of consumer-appealing characteristics, where z is the number of considered characteristics. In our scenario, $\mathbf{y}$ represents a sensorial property with $z = 8$ sampled values over time.

We model the problem of predicting the outcomes of a formulation as a multi-regression task on the function $F : \mathbb{R}^d \mapsto \mathbb{R}^z$ that maps formulations to their consumer-appealing properties. Hence, the objective of the DGM-based model is to solve a multi-regression task and learn an optimal underlying latent graph for solving the task.

Initially, the model takes the input feature matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$ and generates a graph $\mathcal{G}$ as its output. The process involves transforming input features $\mathbf{X} \in \mathbb{R}^{N \times d}$ into auxiliary features $\hat{\mathbf{X}} = f_{\boldsymbol{\Theta}}(\mathbf{X}) \in \mathbb{R}^{N \times \hat{d}}$ using a parametric function $f_{\boldsymbol{\Theta}}$. Then, the auxiliary features $\hat{\mathbf{X}}$ are utilized for graph construction. In

particular, the auxiliary features are used to construct a matrix $\mathbf{P} \in \mathbb{R}^{N \times N}$, where each element p_{ij} represents the probability of an edge between formulations i and j. Afterward, the probability matrix is leveraged to construct the adjacency matrix $\mathbf{A}$ of the graph $\mathcal{G}$. The edge probabilities are defined as follows:

$$p_{ij}(\mathbf{X}; \boldsymbol{\Theta}, t) = e^{-t\Delta(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)^2} = e^{-t\Delta(f_{\boldsymbol{\Theta}}(\mathbf{x}_i), f_{\boldsymbol{\Theta}}(\mathbf{x}_j))^2}, \tag{1}$$

Here, t is a learnable parameter, and $\Delta(\cdot, \cdot)$ represents the Euclidean distance between two nodes in the graph embedding space.

Once the probability matrix $\mathbf{P}(\mathbf{X}; \boldsymbol{\Theta}, t)$ is obtained, a graph $\mathcal{G}$ is derived by constructing a sparse k-degree graph using the k-NN rule, as detailed in [8], obtaining the unweighted adjacency matrix $\mathbf{A}(\mathbf{X}; \boldsymbol{\Theta}, t)$.

Given the adjacency matrix $\mathbf{A}$, the second component of the model takes in input $\mathbf{A}$ and the initial features $\mathbf{X}^{(\mathbf{0})}$, yielding a new set of features $\mathbf{X}^{(1)} = g_{\boldsymbol{\Phi}}(\mathbf{X}^{(0)})$ as output, where $g_{\boldsymbol{\Phi}}$ represents a graph neural network function on $\mathbf{X}$ and $\mathbf{A}$. In our model, $g_{\boldsymbol{\Phi}}$ is a one-layer GCN [11].

The final node features $\mathbf{X}^{(1)}$ is then given as input to an MLP to obtain the final node predictions $\tilde{y}_i = \text{MLP}(X^{(1)})$.

The entire DGM-based model is optimized in an end-to-end fashion constructing a compound loss function that provides incentives for edges involved in accurate predictions while imposing penalties for edges with large prediction errors [8]:

$$\mathcal{L}(y_i, \tilde{y_i}) = \sum_{\substack{i \in V \\ j:(i,j) \in E}} \delta(y_i, \tilde{y}_i) \log p_{ij} \tag{2}$$

where V is the set of nodes, E is the set of obtained edges, y_i and $\tilde{y_i}$ the correct and predicted values for node i, p_{ij} is the probability score for edge (i, j), and $\delta(y_i, \tilde{y}_i)$ is a reward function:

$$\delta(y_i, \tilde{y}_i) = \sum_{m=1}^{z} |y_i(m) - \tilde{y_i}(m)| \tag{3}$$

where z is the number of regression task, and the notation $y_i(m)$ indicates the correct value for the product i on property m.

4.2 Explainer Model

The *Explainer* takes the trained DGM-based model and its prediction(s), and it returns an explanation in the form of a small subgraph of the input graph together with a rank of the node features most influential for the prediction(s). Specifically, we adopt the GNNExplainer [23] method as it is the most well-known and consolidated explainability technique for graph neural networks.

GNNExplainer specifies an explanation as a rich subgraph of the entire graph the GNN was trained on, such that the subgraph maximizes the mutual information with GNN's prediction(s). This is achieved by formulating a mean field

variational approximation and learning a real-valued graph mask that selects the important subgraph of the GNN's computation graph. Simultaneously, GNNExplainer also learns a feature mask that masks out unimportant node features.

In our application scenario, it is important to provide both local, i.e. related to single nodes, and global-level explanations. Global-level explanations are useful for identifying qualitatively the most important characteristics for obtaining a desired product property. Hence, we provide global-level explanations by computing global feature importance scores that can be obtained by simply averaging the feature mask learned by GNNExplainer. Alternatively, global feature importance scores can be provided by computing the permutation importance [2] on node features.

On the other hand, local-level explanations are useful for identifying potential starting recipes - nodes - for new formulation trials to test in our platform or real labs, as well as product characteristics - features - that impact the value of the target property. While potential starting recipes can be identified by analyzing the subgraphs learned by GNNExplainer, the feature mask provided by the model can only highlight the subset of the most influential features and it is not able to quantify the impact of the features on the value of the target property. Hence, we compute the local feature importance of the feature i for the node v as the percentage change in the target property of v in response to a 5% change in the value of i. The percentage of change is chosen following the R&D department's recommendations.

4.3 Explore and Simulate Product Development

We visualize the results of the trained machine learning models and test them on new data in a web application. Specifically, once the user has imported tabular data representing the history of its laboratory trials, the application allows them to explore the learned graph representation, obtain local and global-level node explanations, and initiate simulations to predict outcomes of new products. Based on the results of the simulations, users can decide which products to test in real laboratories, and eventually approve their trials to add them as new training instances, updating the trained models in an online learning fashion. The next two paragraphs will detail the *Explorer* and *Simulation* modules.

Explorer. The *Explorer* is the data visualization module that allows users to visualize the complex relationships among their recipes, and the most influential characteristics/previous trials, to determine good starting points for potential new recipes. Figure 2 shows the graphical interface of the *Explorer* component. The graph constructed through the DGM-based model is displayed as an interactive graph on which users can interact to zoom in on details, select nodes, and view explanations and ancillary information. The interactive graph is built using the Cytoscape.js framework [4] and displayed using the Cola layout[1]. The platform offers filtered views for large graphs to ensure that users are directly

[1] https://marvl.infotech.monash.edu/webcola/, March 2024.

presented with the most interesting nodes. Specifically, the application utilizes the PageRank scores [13] to prioritize and suggest the subgraph around the most important nodes first. Below the interactive graph, a horizontal bar chart displays the global feature importance, which identifies the variables that most significantly impact the predictions across the entire network. Users can select a target node u to see its features, view its local explanations, and obtain the subgraph of the top N nodes starting from u using a dept- or a breadth-first search. The graphical interface for the *Explainer* is very similar to the *Explorer*, with the interactive graph representing the subgraph for a node-level explanation, and the bar chart showing its local feature importance.

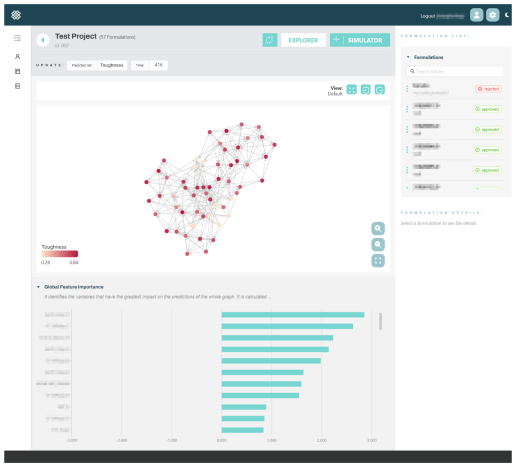


Fig. 2. Example of the GUI for the *Explorer* component in the web application. The graph constructed through the DGM-based model is displayed as an interactive graph. Nodes are colored based on the predicted "toughness" value, the darker the higher. Below the interactive graph, a horizontal bar chart displays the global feature importance for predicting the "toughness".

Simulation. Users can initiate simulations for predicting product outcomes for specific properties from trained machine learning models. First, they can select an existing formulation, chosen with the *Explorer* component, to use as a good basis for a new trial and edit it to create a new recipe. Subsequently, they can run a simulation to obtain the predicted outcome and the relative explanation of the new formulation. Based on the results, users can now edit the recipe multiple times to quickly investigate slight changes in their formulation. At the end of this process, users have the option to save the simulation as a new node within the system. This way, the new node will be part of the training set and the model parameters will be updated. Figure 3 shows an example of the simulation interface for the product property "malleability".

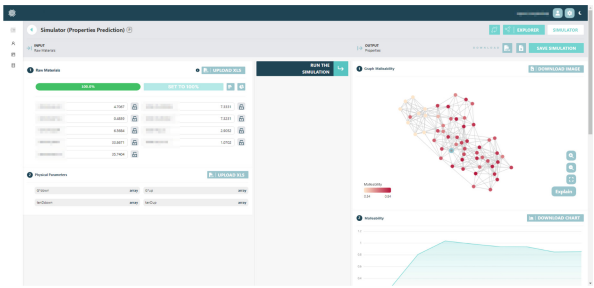


Fig. 3. Example of the GUI for simulations. On the left side, the user can insert and edit a new formulation. On the right side, it can visualize the newly obtained graph and the predicted values for the desired property. The newly inserted formulation is represented by the node circled in light blue. (Color figure online)

5 Experiments

In this section, we compare our DGM-based model against five common baselines for structure learning tasks. Subsequently, we provide insights into the explanation generated on our two datasets. Finally, we validate our method on a benchmark dataset for structure learning.

Experimental Setting. We developed the DGM-based models and the *Explainer* using PyTorch Geometric (PyG)[2]. We evaluated the DGM-based model over the multi-regression task. We used the mean absolute error (MAE) and the root mean squared error (RMSE) to evaluate the prediction performance. We split the datasets into training, validation, and test sets adopting a 60/20/20 split. Consistently, we apply identical dataset divisions and training procedures across all the experiments. In all our experiments, we use the Adam [10] optimizer on the L1 loss on the training set. Hyperparameters are tuned by optimizing the loss function on the validation set, and the model parameters are randomly initialized. More information can be found in the supplementary material.

Baselines. We compare our model against five different baselines. Specifically, following previous works on structure learning [8,9], we choose an MLP that processes the tabular data as is, GCN [11] and GAT [20] using the k-NN graph construction on tabular data, and DGCNN [22], where the graph is dynamically constructed using nearest neighbors in the feature space and learned end-to-end with the supervised task. In addition, we consider a GraphTransformer [19] (GT) baseline, where the graph structure is learned using an attention mechanism over the complete graph. For a fair comparison, we use nearest neighbors with $k = 7$ for all the experiments and we adopt the same embedding dimension for all the models.

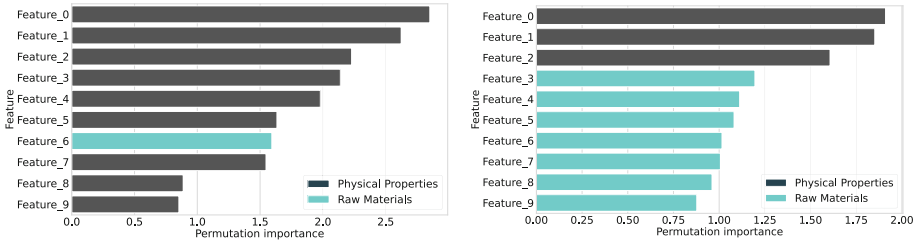
[2] https://www.pyg.org/, June 2024.

Results. We report the RMSE and MAE for "malleability" and "toughness" property prediction in Table 1. For each model, we ran experiments with five different random seeds, reporting the average result and standard deviation for each method. Our model shows better prediction performance and lower variation compared to all the other baselines. Moreover, the GraphTransformer reaches the second-best performance, highlighting that it is a strong baseline to consider in structure learning evaluation. Overall, the results show that models able to learn a latent graph structure perform better than models with a fixed k-NN graph topology. Finally, it is worth noting that an MLP, which does not take into account relationships among items, reaches comparable performances with structure learning models; however, it exhibits large variations in its results.

Table 1. Results for "malleability" and "toughness" prediction on test set using RMSE and MAE. The lower, the better. For each model, we ran experiments with five different random seeds, reporting the average result and standard deviation for each method.

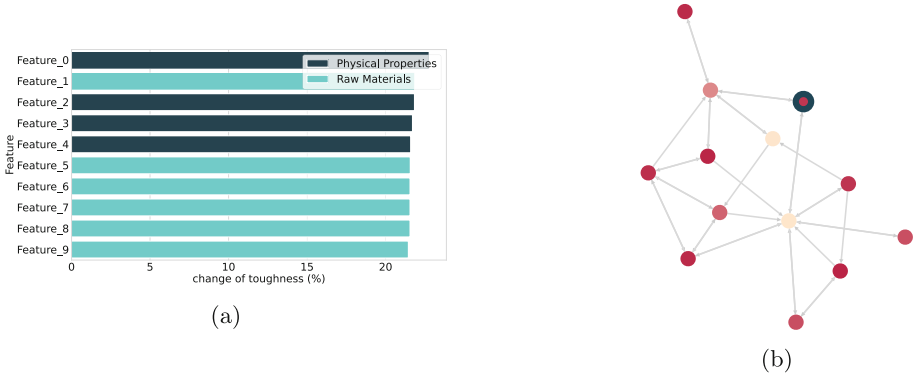
Method	Malleability		Toughness	
	RMSE	MAE	RMSE	MAE
MLP	23.40 ± 18.30	21.60 ± 18.90	28.90 ± 16.20	23.80 ± 17.80
GCN	67.60 ± 03.30	66.50 ± 03.30	70.70 ± 03.00	69.30 ± 03.20
GAT	60.50 ± 09.20	59.10 ± 09.40	62.70 ± 09.40	60.80 ± 09.50
GT	22.30 ± 07.30	20.10 ± 07.00	27.50 ± 03.40	24.80 ± 03.40
DGCNN	30.60 ± 04.70	28.90 ± 04.90	32.60 ± 05.10	30.80 ± 05.40
Our model	**14.22 ± 00.43**	**10.54 ± 00.46**	**12.48 ± 00.73**	**09.15 ± 00.29**

Feature and Subgraph Importance. We report the top ten globally important features for the property "malleability" and "toughness" in Fig. 4. Feature names are anonymized for trade secret protection. We use two different colors for raw materials and physical features. Results show that "toughness" is more influenced by the choice of ingredients while "malleability" is heavily affected by the physical properties of the final product. As an example, given the node with the highest PageRank score, we report its local explanations in Fig. 5. Nodes are colored based on their "toughness": the darker, the higher. Results allow the R&D department to identify new recipes and do the restock of raw materials. In fact, the local feature importance allows the users to understand which recipe changes have an impact on the outcome of the product, while the subgraph gives an idea of the effect of recipe changes by exploring the relationships and comparing the formulations. It is worth noting that local-level explanations are impacting the production process of the products since the company has reduced up to 30% the waste of material.



(a) (b)

Fig. 4. Global feature importance for the "malleability" (4a) and "toughness" (4b) properties. Features are anonymized for trade secret protection. Raw material and physical properties are colored in cyan and dark blue, respectively. "toughness" is more influenced by the choice of ingredients, while "malleability" is heavily affected by the physical properties of the final product. (Color figure online)



(a) (b)

Fig. 5. Local explanations for the "toughness" property of a target node. (a) The top ten locally important features for the node (anonymized for trade secret protection). (b) The important subgraph for its "toughness" prediction. The target node is circled in dark blue. Nodes are colored based on their "toughness": the darker, the higher. The local-level explanations reduce up to 30% the waste of material of the company. (Color figure online)

Ablation Study. We conducted an ablation study of our model by removing the following components: the preprocessing MLP layers, the structure learning (SL) module, and the GNN layer. We report the prediction performances for the property "toughness" in Table 2a. Results show that the graph machine learning approach - the use of structure learning and GNN layers - plays a crucial role in obtaining the best performance against the baselines. Moreover, the use of an MLP to preprocess the features is beneficial in improving the performance of the model and reducing its variability.

Table 2. (2a) Ablation study of our model considering the performance on the test set for the "toughness" property. (2b) Impact of the GNN architecture for the "toughness" property. L is the number of layers.

(a)

Method	RMSE
Our model	**12.48 ± 00.73**
Model w/o MLP	14.10 ± 03.05
Model w/o SL	28.07 ± 01.91
Model w/o GNN	28.90 ± 16.20

(b)

Method	RMSE
GAT (L=1)	13.04 ± 0.65
GCN (L=1)	12.48 ± 0.73
GCN (L=2)	12.01 ± 0.36
GCN (L=3)	12.30 ± 0.29

Choice of GNN Architecture. As described in Sect. 4, our model employs only one GCN layer to process the graph constructed by DGM. In this way, we leverage the relationships between formulations learned by the model avoiding aggregating features of too dissimilar recipes. To show the effectiveness of our choice, we compare the result of our model using one GAT layer and two or three GNN layers in Table 2b. Results show that there is no substantial gain in using more than one GCN layer and that an attention mechanism is not beneficial for the downstream task.

Model Validation. Besides our application scenario, to facilitate transparency and increase trust, we validate our model on a standard benchmark dataset for structure learning tasks in the field of healthcare and brain imaging. Specifically, we utilize the TadPole dataset [16], which contains multimodal data related to 564 patients. The task is the classification of the patients into three classes: "Normal Control", "Alzhaimer's disease" and "Mild cognitive impairment", which represent their clinical status. Each patient is represented by a 354-dimensional representation derived from imaging (MRI, fMRI, PET) and non-imaging (demographics and genotypes) features. We follow the experimental setting used in Kazi *et al.* (IA-GCN) [9] and we compare our model with the same baselines as, to the best of our knowledge, it represents the most updated and recent benchmark evaluation on the TadPole dataset. We report the results of our experiments in Table 3. Note that cDGM stands for continuous DGM, which is a model that utilizes a graph construction technique that leads to a dense network, as described in [8], and it is different from the strategy used by our solution. The results show that our model reaches the best performance against the state-of-the-art models for the task.

Table 3. Performance of our DGM-based model compared with the results presented in IA-GCN on the Tadpole dataset. Results of the baselines were taken from [9]. LC stands for Linear Classifier. cDGM stands for continuous DGM.

Method	Accuracy	AUC	F1
LC	70.22 ± 06.32	80.26 ± 04.81	68.73 ± 06.70
GCN	81.00 ± 06.40	74.70 ± 04.32	78.4 ± 06.77
GAT	81.86 ± 05.80	91.76 ± 03.71	80.90 ± 05.80
DGCNN	84.59 ± 04.33	83.56 ± 04.11	82.87 ± 04.27
cDGM	92.92 ± 02.50	97.16 ± 01.32	91.4 ± 03.32
IA-GCN	96.08 ± 02.49	98.6 ± 01.93	94.77 ± 04.05
Our model	**97.18 ± 00.72**	**99.02 ± 00.62**	**96.40 ± 00.62**

6 Conclusion

In this work, we propose a data-driven approach for fast product development, allowing companies to avoid an intensive trial-and-error iterative process for creating a new or improved product. Starting from tabular formulations of past experiments conducted by a research and development department, a machine-learning model is trained to predict the desired properties of unseen formulations. Specifically, we utilized a graph machine learning model that can simultaneously learn the prediction task and an underlying latent graph to explore similarities and complex relationships between the experiments. Subsequently, explainability techniques based on graphs, perturbation, and sensitivity analysis effectively support R&D in identifying new recipes for reaching a desired property. The constructed graph, the property predictions, their explanations, and the model at inference time are offered to the R&D department through data exploration, visualization, and editing modules in a web application. We presented a case study in the context of food design where industrial researchers are interested in finding recipes that satisfy specific sensorial properties of the product over time. We show the effectiveness of our model on two datasets derived from the case study, achieving the best performance compared to other tabular, graph, and structure learning approaches. Explainability techniques allow users to identify the most influential characteristics for obtaining a certain property, discover new recipes to test in real laboratories and do the restock of raw materials. Thanks to our solutions, the company is expected to reduce the labor-intensive experiments from 150 tests before approval to 30, the waste of materials dropped up to 30%, and the time to market has passed from seven to two months.

In future works, we plan to apply our methodology to additional case studies in several application scenarios that exhibit huge collections of past trials such as drug design or insurance stipulation.

Acknowledgments. We would like to thank the members of the PVM Lainate team for the opportunity given to us by this collaboration, which fostered the development of

this novel methodology. A special thanks to Marco Violi and Maria Giovanna Esposito for their time and assistance in understanding and analyzing the data.

References

1. Material prediction for design automation using graph representation learning. In: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. Volume 3A: 48th Design Automation Conference (DAC) (08 2022). https://doi.org/10.1115/DETC2022-88049, https://doi.org/10.1115/DETC2022-88049
2. Altmann, A., Toloşi, L., Sander, O., Lengauer, T.: Permutation importance: a corrected feature importance measure. Bioinformatics **26**(10), 1340–1347 (2010). https://doi.org/10.1093/bioinformatics/btq134
3. Dileo, M., Zignani, M., Gaito, S.: Temporal graph learning for dynamic link prediction with text in online social networks. Mach. Learn. **113**(6), 1–20 (2023). https://doi.org/10.1007/s10994-023-06475-x
4. Franz, M., et al.: Cytoscape.js 2023 update: a graph theory library for visualization and analysis. Bioinformatics **39**(1), btad031 (2023)
5. Gaudelet, T., et al.: Utilizing graph machine learning within drug discovery and development. Briefings Bioinf. **22**(6), bbab159 (2021). https://doi.org/10.1093/bib/bbab159
6. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: ICML Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272. PMLR (2017)
7. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: LightGCN: simplifying and powering graph convolution network for recommendation. In: SIGIR, pp. 639–648. ACM (2020)
8. Kazi, A., Cosmo, L., Ahmadi, S., Navab, N., Bronstein, M.M.: Differentiable graph module (DGM) for graph convolutional networks. IEEE Trans. Pattern Anal. Mach. Intell. **45**(2), 1606–1617 (2023)
9. Kazi, A., Farghadani, S., Aganj, I., Navab, N.: IA-GCN: interpretable attention based graph convolutional network for disease prediction. In: Cao, X., Xu, X., Rekik, I., Cui, Z., Ouyang, X. (eds.) Machine Learning in Medical Imaging: 14th International Workshop, MLMI 2023, Held in Conjunction with MICCAI 2023, Vancouver, BC, Canada, October 8, 2023, Proceedings, Part I, pp. 382–392. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-45673-2_38
10. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015). http://arxiv.org/abs/1412.6980
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (Poster). OpenReview.net (2017)
12. Krahe, C., Bräunche, A., Jacob, A., Stricker, N., Lanza, G.: Deep learning for automated product design. Procedia CIRP **91**, 3–8 (2020). https://doi.org/10.1016/j.procir.2020.01.135, https://www.sciencedirect.com/science/article/pii/S2212827120307769, enhancing design through the 4th Industrial Revolution Thinking

13. Langville, A.N., Meyer, C.D.: A survey of eigenvector methods for web information retrieval. SIAM Rev. **47**(1), 135–161 (2005). https://doi.org/10.1137/S0036144503424786
14. Li, C.T., Tsai, Y.C., Liao, J.C.: Graph neural networks for tabular data learning. In: 2023 IEEE 39th International Conference on Data Engineering (ICDE), pp. 3589–3592 (2023). https://doi.org/10.1109/ICDE55515.2023.00275
15. Li, F., et al.: Developing a dynamic graph network for interpretable analysis of multi-modal MRI data in parkinson's disease diagnosis. In: EMBC, pp. 1–4. IEEE (2023)
16. Marinescu, R.V., et al.: TADPOLE challenge: accurate alzheimer's disease prediction through crowdsourced forecasting of future data. Predict. Intell. Med. **11843**, 1–10 (2019)
17. Merchant, A., Batzner, S., Schoenholz, S.S., Aykol, M., Cheon, G., Cubuk, E.D.: Scaling deep learning for materials discovery. Nature **624**(7990), 80–85 (2023). https://doi.org/10.1038/s41586-023-06735-9
18. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Networks **20**(1), 61–80 (2009). https://doi.org/10.1109/TNN.2008.2005605
19. Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., Sun, Y.: Masked label prediction: Unified message passing model for semi-supervised classification. In: IJCAI, pp. 1548–1554. ijcai.org (2021)
20. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (Poster), OpenReview.net (2018)
21. Verdone, A., Scardapane, S., Panella, M.: Explainable spatio-temporal graph neural networks for multi-site photovoltaic energy production. Appl. Energy **353**, 122151 (2024). https://doi.org/10.1016/j.apenergy.2023.122151, https://www.sciencedirect.com/science/article/pii/S0306261923015155
22. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Trans. Graph. **38**(5), 146:1–146:12 (2019)
23. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: GNNExplainer: generating explanations for graph neural networks. In: NeurIPS, pp. 9240–9251 (2019)
24. Zhu, Y., et al.: A survey on graph structure learning: Progress and opportunities (2022)