

Supplementary Materials for Benchmarking Machine Learning Methods for Synthetic Lethality Prediction in Cancer

1 Matrix factorization-based methods

1.1 SL²MF

Liu et al. proposed the SL²MF¹ model for predicting SL interactions, which is based on matrix factorization. Specifically, this model employs logistic matrix factorization (LMF) to map genes into a low-dimensional latent space and model the probabilities of SL interactions between genes using their latent representations. The matrix $\mathbf{U} \in \mathbb{R}^{n \times d}$ represents the latent vectors of each gene $\mathbf{U}_i \in \mathbb{R}^{1 \times d}$, where d is the dimensionality of the latent space. The probability of an SL interaction between two genes g_i and g_j is calculated using the following logistic function:

$$p_{ij} = \frac{1}{1 + \exp(-\mathbf{U}_i \mathbf{U}_j^T)} \quad (1)$$

In order to improve the predictive accuracy of the model, the authors integrated knowledge from the PPI network and GO annotation into their approach. The authors hypothesized that genes with similar functional and/or network properties should exhibit similar representations in the latent space. To incorporate this information, they utilized additional loss functions in the model that take into account the GO and PPI similarities between genes for SL interaction prediction.

$$L_{\mathcal{G}} = \frac{1}{2} \sum_{i=1}^n \sum_{u_j \in N^G(u_i)} s_{ij}^G \|\mathbf{U}_i - \mathbf{U}_j\|_2^2 \quad (2)$$

$$L_{\mathcal{P}} = \frac{1}{2} \sum_{i=1}^n \sum_{u_j \in N^P(u_i)} s_{ij}^P \|\mathbf{U}_i - \mathbf{U}_j\|_2^2 \quad (3)$$

where N^G and N^P represent the nearest neighbors of each gene in the GO and PPI similarity matrices, respectively.

1.2 GRSMF

Huang et al. proposed a graph regularized self-representative matrix factorization (GRSMF) model², which is based on the self-representative matrix factorization (SMF)³ method. In traditional SMF models, the input data \mathbf{X} is self-represented by a linear combination of its columns as $\mathbf{X} \approx \mathbf{XU}$, where $\mathbf{U} \in \mathbb{R}^{n \times n}$ is the coefficient matrix that represents the columns of \mathbf{X} . Since the SL interaction matrix is symmetric, the representation of its rows should be the same as the representations of its columns, and thus the self-representation of the symmetric matrix \mathbf{X} is given by $\mathbf{X} \approx \mathbf{U}^T \mathbf{XU}$. For the i th gene g_i , \mathbf{U}_{li} denotes the probability of gene g_i being represented by g_l , which captures the similarity between g_i and g_l based on their SL interactions with other genes. By taking into account some constraints, the following regularized self-representative matrix factorization model can be built

$$\min_{\mathbf{U}} \|\mathbf{X} - \mathbf{U}^T \mathbf{XU}\|_F^2 + \lambda \|\mathbf{U}\|_F^2 \quad (4)$$

$$s.t. \quad 0 \leq \mathbf{U} \leq 1, \quad \sum_{l=1}^n \mathbf{U}_{li} = 1, \quad for \quad i = 1, \dots, n, \quad (5)$$

where $\|\cdot\|_F$ is the Frobenius norm and λ is a tuning parameter which controls the influence of the l_2 regularization.

Due to the limited number of known SL interactions, it can be challenging to learn a comprehensive representation matrix. To address this limitation, the authors incorporated prior information that captures similarities among genes, by introducing a graph regularization term to the SMF model. Specifically, let $\mathbf{S} \in \mathbb{R}^{n \times n}$ denote the GO similarity matrix, where \mathbf{S}_{ij} represents the functional similarity between genes g_i and g_j . The graph regularization based on \mathbf{S} is defined as follows:

$$R = \frac{1}{2} \sum_{l=1}^n \sum_{i,j=1}^n \|\mathbf{U}_{il} - \mathbf{U}_{jl}\|^2 \mathbf{S}_{ij} = \text{Tr}(\mathbf{U}^T \mathbf{L} \mathbf{U}) \quad (6)$$

where $\text{Tr}(\cdot)$ denotes the trace of a matrix, \mathbf{D} is a diagonal matrix with $d_{ii} = \sum_{j=1,n} \mathbf{S}_{ij}$ and $\mathbf{L} = \mathbf{D} - \mathbf{S}$. Above all, the final objective optimization function of GRSMF is formulated as

$$\min_{\mathbf{U}} \|\mathbf{X} - \mathbf{U}^T \mathbf{X} \mathbf{U}\|_F^2 + \lambda \|\mathbf{U}\|_F^2 + \beta \text{Tr}(\mathbf{U}^T \mathbf{L} \mathbf{U}) \quad (7)$$

$$\text{s.t. } 0 \leq \mathbf{U} \leq 1, \quad \sum_{l=1}^n \mathbf{U}_{li} = 1, \quad \text{for } i = 1, \dots, n. \quad (8)$$

where β controls the effect of graph regularization.

1.3 CMF-W

Biological data sets are usually represented as matrices that include paired relationship data between two entities. Matrix sets can have many relationships between organisms. Collective matrix factorization (CMF)⁴ and its extension are models designed to collectively learn numerous such connections. Liany et al. proposed a new method called CMF-W⁵, which is an extension of the CMF approach. Specifically, the authors introduced three improvement measures:

First, generate the feature vector of the matrix using principal component analysis (PCA). When performing dimensionality reduction using PCA, the authors selected the minimum number of principal components required to achieve a cumulative explained variance ratio greater than 0.9 as the dimension of the reduced matrix. Second, provide the graph-based features that are matrix-specific. Including node degree, compactness centrality, intermediate centrality⁶, information centrality⁷, feature vector centrality⁸, Gil Schmidt power index⁹ and Flow Between score¹⁰. Third, incorporating the matrix-specific weight matrix.

In the CMF-W method, each matrix is modeled as the product of three factors:

$$\mathbf{X}^{(m)} \approx \mathbf{U}^{(r_m)} \mathbf{U}^{(c_m)^T} \mathbf{W}^{(m)} \quad (9)$$

The first two items are consistent with the original CMF method, that is, the row and column entity representation of the matrix, and the third item is the matrix specific weight matrix $\mathbf{W}^{(m)}$. For two matrices with the same row and column potential factors, $\mathbf{W}^{(m)}$ can be different.

The latent factors (including $\mathbf{W}^{(m)}$) are learned by solving the optimization problem:

$$\min_{\mathbf{U}, \mathbf{W}} \sum_{m=1}^M d \left(\mathbf{X}^{(m)}, \mathbf{U}^{(r_m)} \mathbf{U}^{(c_m)^T} \mathbf{W}^{(m)} \right) \quad (10)$$

where d is the Frobenius norm of the difference between $\mathbf{X}^{(m)}$ and $\mathbf{U}^{(r_m)} \mathbf{U}^{(c_m)^T} \mathbf{W}^{(m)}$. For $m \times n$ matrix $\mathbf{X}^{(m)}$ and latent dimension k , the dimensions of $\mathbf{U}^{(r_m)}$, $\mathbf{U}^{(c_m)^T}$, $\mathbf{W}^{(m)}$ are $m \times k$, $n \times k$, $n \times n$, respectively.

CMF-W overcomes the issue that arises when several input matrices include the same entity type, the classic CMF cannot learn the unique representation of each entity. Furthermore, the revised model can be directly utilized for input data and derived features, minimizing the effort of feature engineering.

2 Graph neural network methods

2.1 DDGCN

Due to the extreme sparsity of known SL interactions, overfitting is a common issue when training on such a sparse graph. While dropout is usually used to prevent overfitting, the effect of traditional dropout may be inadequate for SL interaction prediction. To enhance the robustness of gene representation for SL prediction, Cai et al. proposed a more effective and robust double dropout mechanism based on GCN, called DDGCN¹¹. This model is composed of two paths with shared parameters that model the dual forms of dropout, namely coarse-grained node dropout and fine-grained edge dropout. The DDGCN model is designed to improve the accuracy of SL interaction prediction by enabling more robust gene representation.

Coarse-grained node dropout is achieved by using the identity matrix \mathbf{I} . Dropout is applied to the identity matrix, and subsequently encode the first-layer representation of each gene using graph convolution:

$$\mathbf{H}_{\mathbf{I}}^{(1)} = \text{ReLU} \left(\hat{\mathbf{A}} \left(\mathbf{M}_{\mathbf{I}}^{(1)} \odot \mathbf{I} \right) \mathbf{W}^{(1)} \right), \quad (11)$$

where \odot is the element-wise multiplication. $\mathbf{M}_\mathbf{I}^{(1)} \in \mathbb{R}^{n \times n}$ is the dropout mask in the first layer. $\mathbf{W}^{(1)} \in \mathbb{R}^{n \times d_1}$ is a trainable weight matrix for the first layer. $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \tilde{\mathbf{A}} \mathbf{D}^{-\frac{1}{2}}$ is the normalized graph adjacency matrix following Kipt et al.¹², where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, \mathbf{A} is the adjacency matrix and $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix which the diagonal elements are defined as $d_{ii} = \sum_{j=1}^n \tilde{\mathbf{A}}_{ij}$.

Fine-grained edge dropout is achieved by using the adjacency matrix \mathbf{A} , where the i th row or column of \mathbf{A} corresponds to the SL interactions between gene g_i and others. Authors apply dropout to \mathbf{A} as follows

$$\mathbf{H}_\mathbf{A}^{(1)} = \text{ReLU} \left(\hat{\mathbf{A}} \left(\mathbf{M}_\mathbf{A}^{(1)} \odot \mathbf{A} \right) \mathbf{W}^{(1)} \right) \quad (12)$$

$$\mathbf{H}_\mathbf{A}^{(2)} = \hat{\mathbf{A}} \left(\mathbf{M}_\mathbf{A}^{(2)} \odot \mathbf{H}_\mathbf{A}^{(1)} \right) \mathbf{W}^{(2)} \quad (13)$$

where $\mathbf{M}_\mathbf{A}^{(l)}$ is the dropout mask in the l th layer.

An important aspect of this model is parameter sharing, which allows the GCN parameters $\mathbf{M}^{(l)}$ in each layer to be shared by the dual dropouts.

The DDGCN model generates two embedding matrices $\mathbf{H}_\mathbf{I}^{(2)}$ and $\mathbf{H}_\mathbf{A}^{(2)}$ through the two paths. To estimate the interaction confidence of each gene pair (g_i, g_j) , the following formula will be used:

$$\hat{y}_\mathbf{I}(i, j) = \text{Dec} \left(\mathbf{H}_\mathbf{I}^{(2)} \right) \quad (14)$$

$$\hat{y}_\mathbf{A}(i, j) = \text{Dec} \left(\mathbf{H}_\mathbf{A}^{(2)} \right) \quad (15)$$

where $\text{Dec}(\cdot)$ is the inner-product decoder¹³ defined as

$$\text{Dec}(\mathbf{U}) = \sigma \left(\mathbf{U} \mathbf{U}^\top \right) = \frac{1}{1 + \exp(-\mathbf{U} \mathbf{U}^\top)} \quad (16)$$

Let $y_{ij} = 1$, if (g_i, g_j) is a known SL interaction, otherwise $y_{ij} = 0$. Thus, the overall loss function can be formulated as follows:

$$L = \sum_{i=1}^n \sum_{j=i+1}^n \text{CE}(y_{ij}, \hat{y}_\mathbf{I}(i, j)) + \alpha \text{CE}(y_{ij}, \hat{y}_\mathbf{A}(i, j)), \quad (17)$$

where CE is the cross entropy loss, $\alpha > 0$ is a hyperparameter controlling the trade-off between the two paths. For the final prediction, they use the geometric mean to aggregate the two prediction scores $\hat{y}_\mathbf{I}(i, j)$ and $\hat{y}_\mathbf{A}(i, j)$:

$$\hat{y}(i, j) = \sqrt[1+\alpha]{\hat{y}_\mathbf{I}(i, j) \times \hat{y}_\mathbf{A}(i, j)}^\alpha \quad (18)$$

2.2 MGE4SL

In order to tackle the sparsity of SL data, Lai et al.¹⁴ suggested a multi graph integrated network structure, and added seven additional biological features by integrating other gene relational databases (such as GO, Corum, Reactome, etc.) to improve the performance of SL prediction.

For each graph, a two-layer GCN is used to generate the embeddings of the nodes. The layers are formulated as follows:

$$\mathbf{H}_\mathbf{A}^{(1)} = \text{ReLU} \left(\hat{\mathbf{A}} \mathbf{X} \mathbf{W}^{(1)} + \mathbf{B} \right) \quad (19)$$

$$\mathbf{H}_\mathbf{A}^{(2)} = \hat{\mathbf{A}} \mathbf{H}_\mathbf{A}^{(1)} \mathbf{W}^{(2)} + \mathbf{B} \quad (20)$$

where the $\hat{\mathbf{A}}$ is defined in the same way as in the previous DDGCN.

Since the data from Corum were too sparse, the authors eventually used the other six graphs to obtain the multi-type node embeddings separately, and then they separately tried summing, splicing, and using CNNs to fuse these features and use them to predict SL interactions. Among these, concat method has better results. The author believes that this is because concat method keeps more knowledge information, and a simpler fully connected network is more successful than a complex CNN network.

2.3 GCATSL

Long et al. proposed GCATSL¹⁵, which is a graph context attention network based on GAT¹⁶ that utilizes multiple biological data sources to generate diverse feature maps as input. The authors introduced a dual attention mechanism, operating at both the node and feature level, to capture the influence of local and global neighbors for learning gene expression from different feature maps. The extracted features were aggregated with the original features using a multilayer perceptron.

Local representations are obtained from the set of nodes that are directly connected to gene g_i , which are defined as its local neighbors. The authors introduce a node-level attention mechanism to learn the different levels of importance of nodes in local representations. Specifically, for a given gene g_i , the importance of its local neighbors is calculated as follows:

$$e_{ij}^l(g_i, g_j) = f(\mathbf{W}_1 h_i, \mathbf{W}_1 b_j), \quad (21)$$

where e_{ij}^l is the attention score that indicates the importance of the neighbor g_j to g_i . $f(\cdot)$ denotes a single-layer feed-forward neural network and $\mathbf{W}_1 \in \mathbb{R}^{d_1 \times d_2}$ is a learnable weight matrix. And h_i is come from the feature matrix $\mathbf{H} \in \mathbb{R}^{n \times d_1}$, which is the GO or PPI similarity score matrix obtained by dimensionality reduction using PCA.

In order to make the attention scores comparable across different nodes, a softmax function is applied to normalize the scores across all local neighbors (note as \mathcal{N}_i^l) of gene g_i . The resulting normalized score is denoted as α_{ij}^l .

Then, a new local representation h_i^l for g_i was derived by aggregating the representations of its local neighbors according to their attention coefficients. However, the attention of an individual node may be unstable and may introduce noise in the model. To address this issue and reduce the noise, the node attention is extended to multi-head attention by repeating the node attention mechanism K times, and then concatenate the K learned representations into gene g_i 's local representation as follows:

$$h_i^l = \parallel_{k=1}^K \text{ReLU} \left(\sum_{j \in \mathcal{N}_i^l} \alpha_{ij}^l \cdot \mathbf{W}_1^k b_j \right) \quad (22)$$

where \parallel denotes the operation of vector concatenation.

Global representations are obtained from the global neighbors of gene g_i . Global neighbors are defined as nodes that are at least two hops away from a given node in the graph. Intuitively, global neighbors can provide valuable information for modeling the center node. To this end, a random walk with restart (RWR)¹⁷ based attention mechanism was employed to learn the node representations from the global neighbors.

After obtaining the global neighborhood of a node, similar to the local representations, the global representations of a node h_i^g can be obtained by the following equation:

$$h_i^g = \parallel_{k=1}^K \text{ReLU} \left(\sum_{j \in \mathcal{N}_i^g} \alpha_{ij}^g \cdot \mathbf{W}_1^k b_j \right) \quad (23)$$

To combine the representations more accurately, the bi-interaction aggregator is used, which encodes feature interactions through two MLPs. The bi-interaction aggregator is formulated as follows:

$$\begin{aligned} z_i &= \text{LeakyReLU} \left(\mathbf{W}_2 \left(h_i + h_i^l + h_i^g \right) + b_2 \right) \\ &+ \text{LeakyReLU} \left(\mathbf{W}_3 \left(b_i \parallel h_i^l \parallel h_i^g \right) + b_3 \right), \end{aligned} \quad (24)$$

where $\mathbf{W}_2 \in \mathbb{R}^{Kd_2 \times d_3}$, $\mathbf{W}_3 \in \mathbb{R}^{3Kd_2 \times d_3}$, $b_2 \in \mathbb{R}^{d_3}$ and $b_3 \in \mathbb{R}^{d_3}$ are trainable weight operation. z_i is the feature-specific representation of gene g_i .

As different feature graphs contain distinct contextual information, a feature-level attention was implemented to aggregate feature-specific representations. The attention score w_i^k represents the importance of the k th feature graph to the gene g_i .

The final representation \mathbf{Z}_i for g_i is:

$$\mathbf{Z}_i = \sum_{k=1}^T \beta_i^k \cdot z_i^k \quad (25)$$

where T is the number of feature graphs, β_i^k is the normalized attention score using the softmax function. z_i^k is the feature-specific representation of g_i .

The final probability score matrix \mathbf{Q} and reconstruction loss are:

$$\mathbf{Q} = \text{sigmoid}(\mathbf{Z} \mathbf{W}_{de} \mathbf{Z}^T) \quad (26)$$

$$L_{Total} = \sum_{(i,j) \in \Omega^+ \cup \Omega^-} \Phi(\mathbf{Q}_{ij}, \mathbf{A}_{ij}) + \gamma \|\mathbf{W}_{de}\|_F^2 \quad (27)$$

where \mathbf{W}_{de} is a learnable latent factor that projects representations back to original feature space for genes. $\Phi(\cdot)$ is the MSE loss. γ is weight factor that is used to control the impact of \mathbf{W}_{de} .

2.4 SLMGAE

Hao et al. proposed a method for predicting SL interactions using Multi-view Graph Auto-Encoder, named SLMGAE¹⁸. In this approach, the authors utilized graphs from diverse data sources(e.g. PPI, GO, etc.) as support views, and the SL graph as the main view. They applied the GAE to reconstruct the graphs for different views and employed a two-layer GCN to calculate the node embedding matrix \mathbf{Z}_2^m and reconstruct graph \mathbf{S}^m for the main view using the following equations:

$$\mathbf{Z}_1^m = \text{LeakyReLU}(\hat{\mathbf{A}} \mathbf{F} \mathbf{W}_1^m) \quad (28)$$

$$\mathbf{Z}_2^m = \text{LeakyReLU}(\hat{\mathbf{A}} \mathbf{Z}_1^m \mathbf{W}_2^m) \quad (29)$$

$$\mathbf{S}^m = \mathbf{Z}_2^m \mathbf{W}_d^m \mathbf{Z}_2^m^T \quad (30)$$

where the $\hat{\mathbf{A}}$ is defined in the same way as in the previous DDGCN. \mathbf{F} is the initial feature matrix. \mathbf{W}_1^m and \mathbf{W}_2^m are the weight matrices, and σ_1 and σ_2 are the activation functions in the first and second layers. \mathbf{W}_d^m is a view specific trainable matrix in the decoder.

Similarly, for each support view u , the node embedding matrix \mathbf{Z}_2^u and reconstruct graph \mathbf{S}^u can be computed by

$$\mathbf{Z}_1^u = \text{LeakyReLU}(\hat{\mathbf{A}}^u \mathbf{F} \mathbf{W}_1^u) \quad (31)$$

$$\mathbf{Z}_2^u = \text{LeakyReLU}(\hat{\mathbf{A}}^u \mathbf{Z}_1^u \mathbf{W}_2^u) \quad (32)$$

$$\mathbf{S}^u = \mathbf{Z}_2^u \mathbf{W}_d^u \mathbf{Z}_2^u^T \quad (33)$$

Then the reconstruction loss \mathcal{L}_M for the main view and \mathcal{L}_S for the support views can be derived as follows

$$\mathcal{L}_M = \Phi(\mathbf{A}, \mathbf{S}^m) \quad (34)$$

$$\mathcal{L}_S = \sum_u \Phi(\mathbf{A}, \mathbf{S}^u) \quad (35)$$

where Φ is the MSE loss, and \mathbf{A} is the adjacency matrix of SL graph.

In addition, because different data sources may play various roles in SL prediction, the author further designs an edge-level attention mechanism to assign different weights to support views to merge all rebuilt graphs.

Based on the attention scheme, a weighted similarity matrix \mathbf{W}_{supp} can be derived as

$$\mathbf{W}_{supp} = \sum_u^{n_s} a^u \odot \mathbf{S}^u \quad (36)$$

where \odot is the element-wise multiplication and a^u is the normalized weight for the support view u .

And then, the final score matrix \mathbf{S}_P will be derived by combining the main view matrix \mathbf{S}_m and the weighted score matrix \mathbf{W}_{supp} from the support views as follows:

$$\mathbf{S}^P = \mathbf{S}^m + c\mathbf{W}_{supp} \quad (37)$$

where c is a hyper-parameter to control the contribution of \mathbf{W}_{supp} for final prediction. Therefore, the final prediction loss \mathcal{L}_P is

$$\mathcal{L}_P = \sum_u \Phi(\mathbf{A}, \mathbf{S}^P) \quad (38)$$

Combine the reconstruction loss and prediction loss, the overall loss \mathcal{L}_{Total} can be obtained as follows:

$$\mathcal{L}_{Total} = \mathcal{L}_M + \alpha\mathcal{L}_S + \beta\mathcal{L}_P \quad (39)$$

where α and β are hyper-parameters, controlling the contributions from \mathcal{L}_S and \mathcal{L}_P , respectively.

2.5 PTGNN

Long et al. introduced a new Pre-Training Graph Neural Networks-based framework named PT-GNN¹⁹, which is utilized to integrate different data sources of link prediction in biomedical networks. Protein sequences contain rich knowledge, and CNNs are able to learn high-order protein features from their sequences for various applications. Following Lee et al.²⁰ and Nguyen et al.²¹, the authors adopt CNN to encode protein features from sequence data. In particular, they split a protein sequence s_p into a set of overlapping n -gram amino acid segments with r as the size of sliding window. Assume that a n -gram amino acid segment is considered as a word, and each word is represented by a d_1 -dimension feature vector. The feature vectors of all words are denoted by $\mathbf{F}_w \in \mathbb{R}^{N_w \times d_1}$, where N_w denotes the number of all possible words in the dataset, and each row of \mathbf{F}_w is the feature vector of a possible word. The word feature matrix \mathbf{F}_w is set as a trainable parameter matrix, which is randomly initialized and can be updated in the pre-training phase for more accurately capturing the intrinsic features of sequences.

Using a CNN-based sequence encoder, an initial feature matrix $\mathbf{X}_p \in \mathbb{R}^{N_w \times d_1}$ can be obtained. Subsequently, the authors propose a GCN-based interaction graph encoder to learn protein representations with input graph \mathcal{G}_p and initial node feature \mathbf{X}_p . For a node v_i in \mathcal{G}_p , the main purpose of the graph encoder is to learn its representation by iteratively aggregating the representations of its neighbors. The ℓ -th layer of a GCN-based graph encoder is as follows:

$$\mathbf{H}_p^{(\ell)} = \text{ReLU} \left(\tilde{\mathbf{A}}_p \mathbf{H}_p^{(\ell-1)} \mathbf{W}_2^{(\ell-1)} + \mathbf{b}_2^{(\ell-1)} \right) \quad (40)$$

where $\tilde{\mathbf{A}}_p$ is the normalized diagonal adjacency matrix with self-connection, $\mathbf{H}_p^{(\ell-1)}$ denotes the outputs of the model at the $(\ell-1)$ -th layer, which $\mathbf{H}_p^{(0)}$ is defined as the input feature matrix \mathbf{X}_p . $\mathbf{W}_2^{(\ell-1)}$ and $\mathbf{b}_2^{(\ell-1)}$ are trainable weight matrix and bias vector, respectively. After several GCN layers, the representations of proteins $\mathbf{H}_p \in \mathbb{R}^{N_p \times d_3}$ can be adopted, where d_3 denotes the dimension of the protein representations. Similarly, the representations of proteins $\mathbf{H}_g \in \mathbb{R}^{N_g \times d_3}$ from the protein GO graph \mathcal{G}_g can be obtained with node features \mathbf{X}_p .

The prediction scores and loss functions of SL are as follows:

$$\mathbf{P} = \text{ReLU} \left(\mathbf{H} \mathbf{H}^\top \right) \quad (41)$$

$$\mathcal{L} = \sum_{(i,j) \in \Omega^+ \cup \Omega^-} \Phi(\mathbf{P}(i,j), \mathbf{A}(i,j)) + \delta \|\Theta\|_F^2 \quad (42)$$

where \mathbf{P} is the reconstructed score matrix where each element describes the interaction score for a node pair, $\mathbf{H} = \lambda \mathbf{H}_p + (1 - \lambda) \mathbf{H}_g$ is the final representations of proteins obtained from the GCN-based graph encoder, λ a tuning parameter which controls the influence of the representations from different data, Θ is the parameter matrix of the pre-training model. δ is weight factor that is used to control the influence of Θ , $\Phi(\cdot)$ is the MSE loss.

2.6 KG4SL

The knowledge graph (KG) integrates multi-source data through standardized semantics, and can support the mining and analysis of complex associated data through the map reasoning ability. Therefore, Wang et al. proposed a GNN based model, KG4SL²², which integrates the KG into the task of SL prediction. The model is basically formed of three sections.

Gene-specific weighted subgraph. Given an SL-related gene, a weighted subgraph from KG can be constructed. Identifying relevant nodes and determining the weights are two key operations to construct the gene-specific weighted subgraph. For gene g , $N(g)$ denotes the set of neighbors of g , from which k neighbors are screened out and denoted as $P(g)$. In a gene-specific subgraph, different weights for edges can be assigned to describe the importance of the relations. For an SL pair (g_i, g_j) , the weight for an edge $r_{g,g'}$ in g_i 's subgraph is computed by $\omega_{g,g'}^{g_i} = g(\mathbf{g}_j, \mathbf{r}_{g,g'})$, where g is one of the entities in the subgraph of g_i , and $g' \in P(g)$. \mathbf{g}_j and $\mathbf{r}_{g,g'}$ are the feature embedding of gene g_j and relation $r_{g,g'}$, respectively. g is an inner product function.

Aggregation of node representations. For any central entity e in the subgraph of gene g_i , the representations of all its picked neighbors were aggregated to update its own representation. To show the topological neighborhood structure of entity e in the KG, the weighted average combination of e 's neighborhood is computed as follows:

$$\mathbf{e}_{P(e)} = \sum_{e' \in P(e)} \tilde{\omega}_{e,e'}^{e_i} \mathbf{e}' \quad (43)$$

where \mathbf{e} is the representation of entity e , gene g_i and gene g_j are a pair in the SL matrix, and $\tilde{\omega}_{e,e'}^{e_i} = \frac{\exp(\omega_{e,e'}^{e_i})}{\sum_{\hat{e} \in P(e)} \exp(\omega_{e,\hat{e}}^{e_i})}$ is the normalized gene-relation score by applying a softmax function.

After obtaining the picked neighbors' representation $\mathbf{e}_{P(e)}$ of a central entity in one hop, similar to Wang et al.²³, it integrates the entity representation \mathbf{e} in to a single vector to update \mathbf{e} :

$$\mathbf{e}[h+1] = \phi(\mathbf{W}(\mathbf{e}[!h] + \mathbf{e}_{P(e)}) + \mathbf{b}) \quad (44)$$

where \mathbf{W} and \mathbf{b} are the linear transformation weight and bias, respectively, and ϕ is an activation function. After aggregating neighbors' information through H hops, the final feature representation of gene $\hat{\mathbf{g}}_i$ is $\mathbf{e}[!h]$, $\hat{\mathbf{g}}_j$ is obtained in the same way.

SL prediction. Finally, the predicted interaction probability between gene g_i and gene g_j is calculated by $\hat{s}_{i,j} = \phi(f(\hat{\mathbf{g}}_i, \hat{\mathbf{g}}_j))$, where f is the inner product function and ϕ is a sigmoid function, squashing the output to a range between 0 and 1.

The base loss L is computed through cross-entropy of the truth label and the predicted label for the edges, represented as follows:

$$L = \max(\hat{s}_{i,j}, 0) - \hat{s}_{i,j} \times s_{i,j} + \log(1 + \exp(-|\hat{s}_{i,j}|)) \quad (45)$$

where $\hat{s}_{i,j}$ is the predicted label and $s_{i,j}$ is the truth label for the edge.

With an L2-regulator, the final loss is defined as:

$$\min_{\mathbf{W}, \mathbf{A}, \mathbf{b}} \ell = \min_{\mathbf{W}, \mathbf{A}, \mathbf{b}} \sum_{i,j} J + \alpha \|\Gamma\| \quad (46)$$

where $\|\Gamma\| = \frac{\|\mathbf{e}\| + \|\mathbf{r}\| + \|\mathbf{W}\|}{2}$, \mathbf{A} is the trainable weight matrix in which each element represents the gene-relation score and α is a balancing hyper-parameter.

2.7 PiLSL

Liu et al. proposed PiLSL²⁴, a GNN model based on paired interactive learning, to learn the paired interactive representation of two genes with SL interaction. The author claims that most of the current methods are to learn the expression of each gene independently, ignoring the representation of the pairwise interaction between two genes. PiLSL consists of three steps to learn the representations of pairwise interactions.

Pairwise enclosing graph extraction. There are abundant biomedical entities connected to the genes by various relations in SynLethKG and there exists interaction information between two genes, authors focus on the local enclosing graph of each gene pair to capture the gene-gene interaction information. For a given gene pair u and v , the enclosing graph

is defined as $\mathcal{G}_{\text{en}} = \{(u, r, v) \mid u, v \in \mathcal{N}_k(u) \cap \mathcal{N}_k(v), r \in \mathcal{R}\}$, where $\mathcal{N}_k(u) = \{s \mid d(s, u) \leq k\}$ and $\mathcal{N}_k(v) = \{s \mid d(s, v) \leq k\}$, $d(\cdot, \cdot)$ denotes the shortest distance between two elements.

Attentive embedding propagation. To reduce the noisy information and extract biological meaning from the prediction, attentive embedding propagation was adapted to discriminate the importance among the edges in the enclosing graph, which consists of three components: message passing, relation-aware attention and message aggregation.

At the l -th layer, message passing used to combine the representations of the neighbors with the node u ,

$$\mathbf{m}_{\mathcal{N}_u}^{(l)} = \sum_{r=1}^{N_R} \sum_{v \in \mathcal{N}_r(u)} a_{r(u,v)}^{(l)} \mathbf{W}_r^l \mathbf{h}_v^{(l-1)} \quad (47)$$

where N_R is the total number of the relations in SynLethKG, $\mathcal{N}_r(u)$ denotes the set of directly connected neighbors of node u under relation r , \mathbf{W}_r^l is the weighted matrix to transform hidden representation in the l -th layer over relation r , $a_{r(u,v)}^{(l)}$ is the attention weight, controlling how much message being passed from v to u via relation r .

Then, the node representations are updated by node self-representation and message passing from its neighbors as follows:

$$\mathbf{h}_u^{(l)} = \text{ReLU} \left(\mathbf{W}_{\text{self}}^{(l)} \mathbf{h}_u^{(l-1)} + \mathbf{m}_{\mathcal{N}_u}^{(l)} \right), \quad (48)$$

where $\mathbf{W}_{\text{self}}^{(l)}$ is the weight matrix of transforming the node self-representation, $\mathbf{h}^{(l)}$ is the embeddings of all entities in the layer l , $\mathbf{h}^{(0)}$ were initialized by TransE.

Latent and explicit features fusion. Latent features consist of information at the node level and the graph level. At the node level, the representation $\mathbf{h}^{(l)}$ will be obtained by attentive embedding propagation. At the graph level, the representation $\mathbf{h}_{\mathcal{G}_{\text{en}}}^{(l)}$ can be got by the average of all node representations in the enclosing graph. The representation of $\mathbf{h}_{\mathcal{G}_{\text{en}}}^{(l)}$ at layer l is:

$$\mathbf{h}_{\mathcal{G}_{\text{en}}}^{(l)} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathbf{h}_i^{(l)} \quad (49)$$

where \mathcal{V} is the set of nodes in the enclosing graph \mathcal{G}_{en} .

The final predicted interaction probability is $\hat{p}_{uv} = \mathbf{W}_{\text{pred}} \mathbf{h}_{uv}$, where \mathbf{W}_{pred} is the weight matrix of the decoder.

Two kinds of losses are used for optimization:

$$L_{CE} = -\frac{1}{|\tau|} \sum_{(u,v) \in \tau} p_{uv} \log \hat{p}_{uv} + (1 - p_{uv}) \log (1 - \hat{p}_{uv}) \quad (50)$$

$$L_W = \|\mathbf{W}_{\text{all}}\|_2^2 \quad (51)$$

where τ is the sampled training set of gene pairs and p_{uv} denotes true label for the SL interaction, \mathbf{W}_{all} denotes all weight parameters.

The total loss is $L_{\text{Total}} = L_{CE} + \frac{\lambda}{2} L_W$, where λ is the coefficient of L_W .

2.8 NSF4SL

Because the prediction task of SL usually encounters the problem of lacking high-quality negative samples, inspired by the negative-sample-free contrastive learning frameworks, Wang et al. proposed a new SL prediction model, NSF4SL²⁵, which gives priority to the positive samples. NSF4SL first extracts the feature augmentation view of each gene, and then inputs the augmentation features and original features of two genes in the SL gene pair into two branches of the contrastive learning framework respectively. First, they extract a feature-wise augmented view for each gene. Second, they feed augmented features of the two genes in an SL pair into the online branch, and the original features of them into the target branch. Third, they obtain the gene embeddings and maximize the similarity between them.

Feature-wise data augmentation is designed to address the sparsity of the training data. Given a gene training set \mathcal{G}_t , for a gene $g \in \mathcal{G}_t$, they obtain the initial feature of g as $\mathbf{x}_g = \{x_g^{(1)}, \dots, x_g^{(d)}\}$, where x_g^i denotes the i^{th} element of \mathbf{x}_g , and d denotes the dimension of the vector. Then, they select a random subset of \mathbf{x}_g and replace them by the respective

feature-level average values. Specifically, in each iteration of training, they sample an augmented view \mathcal{V}_g for each gene g of length q ,

$$\mathcal{V}_g = u(\mathbf{x}_g), \quad q = \lfloor r \cdot d \rfloor, \quad (52)$$

where $u(\cdot)$ is the uniformly sampling function, $\lfloor \cdot \rfloor$ is the floor function and r is the defined augmented ratio. Next, they obtain the augmented representation $\tilde{\mathbf{x}}_g$ for gene g as follows,

$$\tilde{x}_g^{(i)} = \begin{cases} \bar{x}^{(i)}, & \text{if } x_g^i \in \mathcal{V}_g \\ x_g^{(i)}, & \text{otherwise} \end{cases} \quad (53)$$

where $\bar{x}^{(i)}$ is mean of the i^{th} element over the whole gene set \mathcal{G}_t .

Negative-sample-free architecture is to avoid the negative sampling bias in the models for SL prediction. They employ a non-symmetric two-branch network as the backbone model. Instead of classifying SL and non-SL pairs, this network learns to recognize similarities between input SL pairs. The architecture relies on two parallel branches: they contain an online encoder f_θ and a target encoder f_ξ with the corresponding trained weights θ and ξ , respectively. f_θ and f_ξ share the same architecture but have different weights. One key design is the predictor p_ϕ following the online encoder. As the target encoder is used to output the target SLs, p_ϕ uses the gene embeddings of the same SLs from the online encoder to be close to them.

In the end of training, they adopt contrastive learning loss to maximize the similarity between the representations of two genes in an SL pair. Let $(\mathbf{g}_i, \mathbf{g}_j)$ be the features of an input pair of genes known to have SL relation with each other. The online branch output are

$$\mathbf{g}_i^o = p_\phi(f_\theta(\tilde{\mathbf{g}}_i)) \quad \text{and} \quad \mathbf{g}_j^o = p_\phi(f_\theta(\tilde{\mathbf{g}}_j)), \quad (54)$$

and the target branch output as

$$\mathbf{g}_i^t = f_\xi(\mathbf{g}_i) \quad \text{and} \quad \mathbf{g}_j^t = f_\xi(\mathbf{g}_j). \quad (55)$$

They minimize the following objective function \mathcal{L} to maximize the similarity between the representations of the two genes from the two branches,

$$\mathcal{L} = -\frac{1}{2} (h(\mathbf{g}_i^o, \mathbf{g}_j^t) + h(\mathbf{g}_j^o, \mathbf{g}_i^t)), \quad (56)$$

where $h(\cdot)$ is the inner product function between two representations.

References

1. Liu, Y., Wu, M., Liu, C., Li, X.-L. & Zheng, J. Sl2mf: Predicting synthetic lethality in human cancers via logistic matrix factorization. *IEEE/ACM Transactions on Comput. Biol. Bioinforma.* **17**, 748–757, DOI: [10.1109/tcbb.2019.2909908](https://doi.org/10.1109/tcbb.2019.2909908) (2020).
2. Huang, J., Wu, M., Lu, F., Ou-Yang, L. & Zhu, Z. Predicting synthetic lethal interactions in human cancers using graph regularized self-representative matrix factorization. *BMC Bioinforma.* **20**, DOI: [10.1186/s12859-019-3197-3](https://doi.org/10.1186/s12859-019-3197-3) (2019).
3. Ma, S. *et al.* Self-representative manifold concept factorization with adaptive neighbors for clustering. In *IJCAI International Joint Conference on Artificial Intelligence* (2018).
4. Singh, A. P. & Gordon, G. J. Relational learning via collective matrix factorization. DOI: [10.1145/1401890.1401969](https://doi.org/10.1145/1401890.1401969) (2008).
5. Liany, H., Jeyasekharan, A. & Rajan, V. Predicting synthetic lethal interactions using heterogeneous data sources. *Bioinformatics* **36**, 2209–2216, DOI: [10.1093/bioinformatics/btz893](https://doi.org/10.1093/bioinformatics/btz893) (2020).
6. Freeman, L. C. A set of measures of centrality based on betweenness. *Sociometry* **40**, 35–41, DOI: [10.2307/3033543](https://doi.org/10.2307/3033543) (1977).
7. Stephenson, K. & Zelen, M. Rethinking centrality: Methods and examples. *Soc. Networks* **11**, 1–37, DOI: [https://doi.org/10.1016/0378-8733\(89\)90016-6](https://doi.org/10.1016/0378-8733(89)90016-6) (1989).
8. Bonacich, P. Factoring and weighting approaches to status scores and clique identification. *The J. Math. Sociol.* **2**, 113–120, DOI: [10.1080/0022250X.1972.9989806](https://doi.org/10.1080/0022250X.1972.9989806) (1972).

9. Gil-Mendieta, J. & Schmidt, S. The political network in mexico. *Soc. Networks* **18**, 355–381, DOI: [https://doi.org/10.1016/0378-8733\(95\)00281-2](https://doi.org/10.1016/0378-8733(95)00281-2) (1996).
10. Freeman, L. C., Borgatti, S. P. & White, D. R. Centrality in valued graphs: A measure of betweenness based on network flow. *Soc. Networks* **13**, 141–154, DOI: [https://doi.org/10.1016/0378-8733\(91\)90017-N](https://doi.org/10.1016/0378-8733(91)90017-N) (1991).
11. Cai, R., Chen, X., Fang, Y., Wu, M. & Hao, Y. Dual-dropout graph convolutional network for predicting synthetic lethality in human cancers. *Bioinformatics* **36**, 4458–4465, DOI: [10.1093/bioinformatics/btaa211](https://doi.org/10.1093/bioinformatics/btaa211) (2020).
12. Kipf, T. & Welling, M. Semi-supervised classification with graph convolutional networks. *ArXiv* **abs/1609.02907** (2016).
13. Hamilton, W. L., Ying, R. & Leskovec, J. Representation learning on graphs: Methods and applications. *ArXiv* **abs/1709.05584** (2017).
14. Lai, M. *et al.* Predicting synthetic lethality in human cancers via multi-graph ensemble neural network. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, DOI: [10.1109/embc46164.2021.9630716](https://doi.org/10.1109/embc46164.2021.9630716) (IEEE, 2021).
15. Long, Y. *et al.* Graph contextualized attention network for predicting synthetic lethality in human cancers. *Bioinformatics* **37**, 2432–2440, DOI: [10.1093/bioinformatics/btab110](https://doi.org/10.1093/bioinformatics/btab110) (2021).
16. Velickovic, P. *et al.* Graph attention networks. *stat* **1050**, 10.48550 (2017).
17. Tong, H., Faloutsos, C. & Pan, J.-Y. Random walk with restart: fast solutions and applications. *Knowl. Inf. Syst.* **14**, 327–346, DOI: [10.1007/s10115-007-0094-2](https://doi.org/10.1007/s10115-007-0094-2) (2008).
18. Hao, Z. *et al.* Prediction of synthetic lethal interactions in human cancers using multi-view graph auto-encoder. *IEEE J. Biomed. Heal. Informatics* **25**, 4041–4051, DOI: [10.1109/jbhi.2021.3079302](https://doi.org/10.1109/jbhi.2021.3079302) (2021).
19. Long, Y. *et al.* Pre-training graph neural networks for link prediction in biomedical networks. *Bioinformatics* **38**, 2254–2262, DOI: [10.1093/bioinformatics/btac100](https://doi.org/10.1093/bioinformatics/btac100) (2022).
20. Lee, I., Keum, J. & Nam, H. Deepconv-dti: Prediction of drug-target interactions via deep learning with convolution on protein sequences. *PLOS Comput. Biol.* **15**, e1007129, DOI: [10.1371/journal.pcbi.1007129](https://doi.org/10.1371/journal.pcbi.1007129) (2019).
21. Nguyen, T. *et al.* Graphdta: predicting drug–target binding affinity with graph neural networks. *Bioinformatics* **37**, 1140–1147, DOI: [10.1093/bioinformatics/btaa921](https://doi.org/10.1093/bioinformatics/btaa921) (2021).
22. Wang, S. *et al.* Kg4sl: knowledge graph neural network for synthetic lethality prediction in human cancers. *Bioinformatics* **37**, i418–i425, DOI: [10.1093/bioinformatics/btab271](https://doi.org/10.1093/bioinformatics/btab271) (2021).
23. Wang, H. *et al.* Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. DOI: [10.1145/3292500.3330836](https://doi.org/10.1145/3292500.3330836) (2019).
24. Liu, X. *et al.* Pils: pairwise interaction learning-based graph neural network for synthetic lethality prediction in human cancers. *Bioinformatics* **38**, ii106–ii112, DOI: [10.1093/bioinformatics/btac476](https://doi.org/10.1093/bioinformatics/btac476) (2022).
25. Wang, S. *et al.* NSF4SL: negative-sample-free contrastive learning for ranking synthetic lethal partner genes in human cancers. *Bioinformatics* **38**, ii13–ii19, DOI: [10.1093/bioinformatics/btac462](https://doi.org/10.1093/bioinformatics/btac462) (2022).
26. Paladugu, S. R., Zhao, S., Ray, A. & Raval, A. Mining protein networks for synthetic genetic interactions. *BMC Bioinforma.* **9**, 426, DOI: [10.1186/1471-2105-9-426](https://doi.org/10.1186/1471-2105-9-426) (2008).
27. Pandey, G. *et al.* An integrative multi-network and multi-classifier approach to predict genetic interactions. *PLoS Comput. Biol.* **6**, e1000928, DOI: [10.1371/journal.pcbi.1000928](https://doi.org/10.1371/journal.pcbi.1000928) (2010).
28. Wu, M. *et al.* In silico prediction of synthetic lethality by meta-analysis of genetic interactions, functions, and pathways in yeast and human cancer. *Cancer Informatics* **13s3**, CIN.S14026, DOI: [10.4137/cin.s14026](https://doi.org/10.4137/cin.s14026) (2014).
29. Yin, Z., Qian, B., Yang, G. & Guo, L. Predicting synthetic lethal genetic interactions in breast cancer using decision tree. In *Proceedings of the 2019 6th International Conference on Biomedical and Bioinformatics Engineering*, DOI: [10.1145/3375923.3375933](https://doi.org/10.1145/3375923.3375933) (ACM, 2019).
30. Das, S., Deng, X., Camphausen, K. & Shankavaram, U. Discoversl: an r package for multi-omic data driven prediction of synthetic lethality in cancers. *Bioinformatics* **35**, 701–702, DOI: [10.1093/bioinformatics/bty673](https://doi.org/10.1093/bioinformatics/bty673) (2019).
31. Li, J. *et al.* Identification of synthetic lethality based on a functional network by using machine learning algorithms. *J. Cell. Biochem.* **120**, 405–416, DOI: [10.1002/jcb.27395](https://doi.org/10.1002/jcb.27395) (2019).
32. Benstead-Hume, G. *et al.* Predicting synthetic lethal interactions using conserved patterns in protein interaction networks. *PLOS Comput. Biol.* **15**, e1006888, DOI: [10.1371/journal.pcbi.1006888](https://doi.org/10.1371/journal.pcbi.1006888) (2019).

33. Wu, L.-L. *et al.* Synthetic lethal interactions prediction based on multiple similarity measures fusion. *J. Comput. Sci. Technol.* **36**, 261–275, DOI: [10.1007/s11390-021-0866-2](https://doi.org/10.1007/s11390-021-0866-2) (2021).
34. De Kegel, B., Quinn, N., Thompson, N. A., Adams, D. J. & Ryan, C. J. Comprehensive prediction of robust synthetic lethality between paralog pairs in cancer cell lines. *Cell Syst.* **12**, 1144–+, DOI: [10.1016/j.cels.2021.08.006](https://doi.org/10.1016/j.cels.2021.08.006) (2021).
35. Benfatto, S. *et al.* Uncovering cancer vulnerabilities by machine learning prediction of synthetic lethality. *Mol. Cancer* **20**, DOI: [10.1186/s12943-021-01405-8](https://doi.org/10.1186/s12943-021-01405-8) (2021).
36. Seale, C., Tepeli, Y. & Gonçalves, J. P. Overcoming selection bias in synthetic lethality prediction. *Bioinformatics* DOI: [10.1093/bioinformatics/btac523](https://doi.org/10.1093/bioinformatics/btac523) (2022).
37. Wan, F. *et al.* Exp2sl: A machine learning framework for cell-line-specific synthetic lethality prediction. *Front Pharmacol* **11**, 112, DOI: [10.3389/fphar.2020.00112](https://doi.org/10.3389/fphar.2020.00112) (2020).
38. Lu, X., Chen, G., Li, J., Hu, X. & Sun, F. Magcn: A multiple attention graph convolution networks for predicting synthetic lethality. *IEEE/ACM Transactions on Comput. Biol. Bioinforma.* 1–10, DOI: [10.1109/tcbb.2022.3221736](https://doi.org/10.1109/tcbb.2022.3221736) (2022).
39. Fan, K., Tang, S., Gökbağ, B., Cheng, L. & Li, L. Multi-view graph convolutional network for cancer cell-specific synthetic lethality prediction. *Front. Genet.* **13**, DOI: [10.3389/fgene.2022.1103092](https://doi.org/10.3389/fgene.2022.1103092) (2023).
40. Zhu, Y., Zhou, Y., Liu, Y., Wang, X. & Li, J. Slgnn: Synthetic lethality prediction in human cancers based on factor-aware knowledge graph neural network. *Bioinformatics* (2023).

Table S1. List of classical machine learning methods for SL prediction.

Model & Ref.	Year	Description
Paladugu et al. ²⁶	2008	A SVM system that uses graph-theoretic properties of two proteins in a protein interaction network as input features for prediction of synthetic sick/lethal interactions.
Pandey et al. ²⁷	2010	The author defines a large number of more comprehensive SL independent features, and uses these features to develop a MNMC framework for predicting SL interaction. The system can realize multiple classification processes at the same time.
MetaSL ²⁸	2014	The author proposed an ensemble method for predicting yeast SL, named MetaSL, which integrates the output of 17 genomic and proteomic features and 10 classification methods.
SL ² MF ¹	2018	SL ² MF uses logistic matrix factorization to learn gene representations, which are then used to identify potential SL interactions. The authors design an importance weighting scheme to distinguish known and unknown SL pairs and combine PPI and GO information for the prediction.
GRSMF ²	2019	GRSMF is a method based on graph regularized self-representation matrix factorization (MF). It learns self-representation from known SL interactions and further integrates GO information to predict potential SL interactions.
Yin et al. ²⁹	2019	The author developed a DT based method and used the features based on mutation and copy number variation level to predict gene pairs with SL interaction.
DiscoverSL ³⁰	2019	This is a multi-parameter RF classifier, to predict and visualize synthetic lethality in cancers using multi-omic cancer data.
Li et al. ³¹	2019	Using the selected functional features obtained from GO and KEGG data, the author proposed a method based on RF to predict SL gene pairs.
CMFW ⁵	2020	CMFW is a collective matrix factorization-based method that integrates multiple heterogeneous data sources for SL prediction.
SLant ³²	2019	This is an RF based method, which works by identifying and exploiting conserved patterns in protein interaction network topology both within and across species.
Wu et al. ³³	2021	Multiple types of gene similarity measures are integrated and k-NN algorithm is applied to achieve the similarity-based classification task between gene pairs.
De Kegel et al. ³⁴	2021	By analyzing the genome-wide CRISPR screening and the molecular map of cancer cell lines, the author identified the shared PPI and evolutionary conservation features to predict SL, and developed a classifier based on RF according to these features.
PARIS ³⁵	2021	PARIS is an RF based method, it predicts SL interactions by combining CRISPR viability screens with genomics and transcriptomics data across hundreds of cancer cell lines profiled within the Cancer Dependency Map.
SBSL ³⁶	2022	The authors introduced regularized logistic regression or RF into the selection bias resistant synthetic lethality (SBSL) prediction for each gene pair molecularly characterized from cell lines, cancer patient tissues and healthy donor tissue samples.

Table S2. List of deep learning methods for SL prediction.

Model & Ref.	Year	Description
EXP2SL ³⁷	2020	A method based on semi-supervised neural network uses the cell-specific gene expression profile data obtained in the L1000 project to provide features for predicting human SL interaction.
DDGCN ¹¹	2020	DDGCN is the first graph neural network (GNN)-based method for SL prediction. It uses graph convolutional network (GCN) and known SL interaction matrix as features. The authors use coarse-grained node dropout and fine-grained edge dropout to address the issue of overfitting of GCNs on sparse graphs.
GCATSL ¹⁵	2021	GCATSL proposes a graph contextualized attention network to learn gene representations for SL prediction. The authors use data of GO and PPI to generate a set of feature graphs as model inputs and introduce attention mechanisms at the node and feature levels to capture the influence of neighbors and learn gene expression from different feature graphs.
SLMGAE ¹⁸	2021	SLMGAE is a method for predicting SL interactions by leveraging a multi-view graph autoencoder. The authors incorporate data from PPI and GO as supporting views, while utilizing the SL graph as the main view, and apply a graph autoencoder (GAE) to reconstruct these views.
MGE4SL ¹⁴	2021	MGE4SL is a method based on Multi-Graph Ensemble (MGE) to integrate biological knowledge from PPI, GO, and Pathway. It combines the embeddings of features with different neural networks.
KG4SL ²²	2021	KG4SL is a novel model based on graphical neural networks (GNN), and the first method that utilizes knowledge graph (KG) for SL prediction. The integration of KG helps the model obtain more information.
PiSL ²⁴	2021	PiSL is a graph neural network (GNN)-based method that predicts SL by learning the representation of pairwise interaction between two genes.
PTGNN ¹⁹	2021	PTGNN is a pre-training method based on graph neural networks that can integrate various data sources and leverage the features obtained from graph-based reconstruction tasks to initialize models for downstream link prediction tasks.
NSF4SL ²⁵	2022	NSF4SL is a contrastive learning-based model for SL prediction that eliminates the need for negative samples. It frames the SL prediction task as a gene ranking problem and utilizes two interacting neural network branches to learn representations of SL-related genes, thereby capturing the characteristics of positive SL samples.
MAGCN ³⁸	2022	A multiple attention graph convolutional network for predicting synthetic lethality. The authors used GCN to accumulate knowledge from topological structures and genetic features obtained from PPI, GO, and constructed a multi-graph attention model to learn the contributing factors of different graphs and aggregate these graphs to generate embeddings.
MVGCN-iSL ³⁹	2023	A multi-view graph convolutional network (GCN) model to predict cancer cell-specific SL gene pairs by integrating five biogram features and multi-omics data.
SLGNN ⁴⁰	2023	A GNN model that enables a better understanding of the underlying biological mechanisms predicting SL, which models the preferences of genes with different relationships in the knowledge graph, ultimately providing better interpretability.

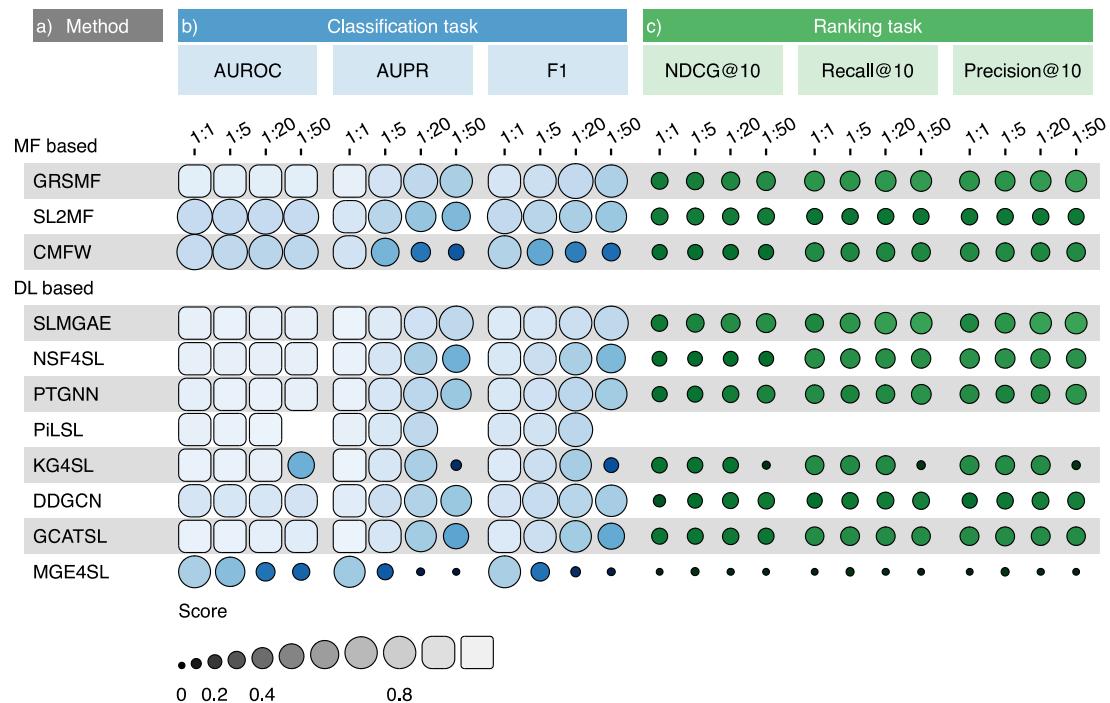


Figure S1. Result under CV1 and NSM_{Rand}

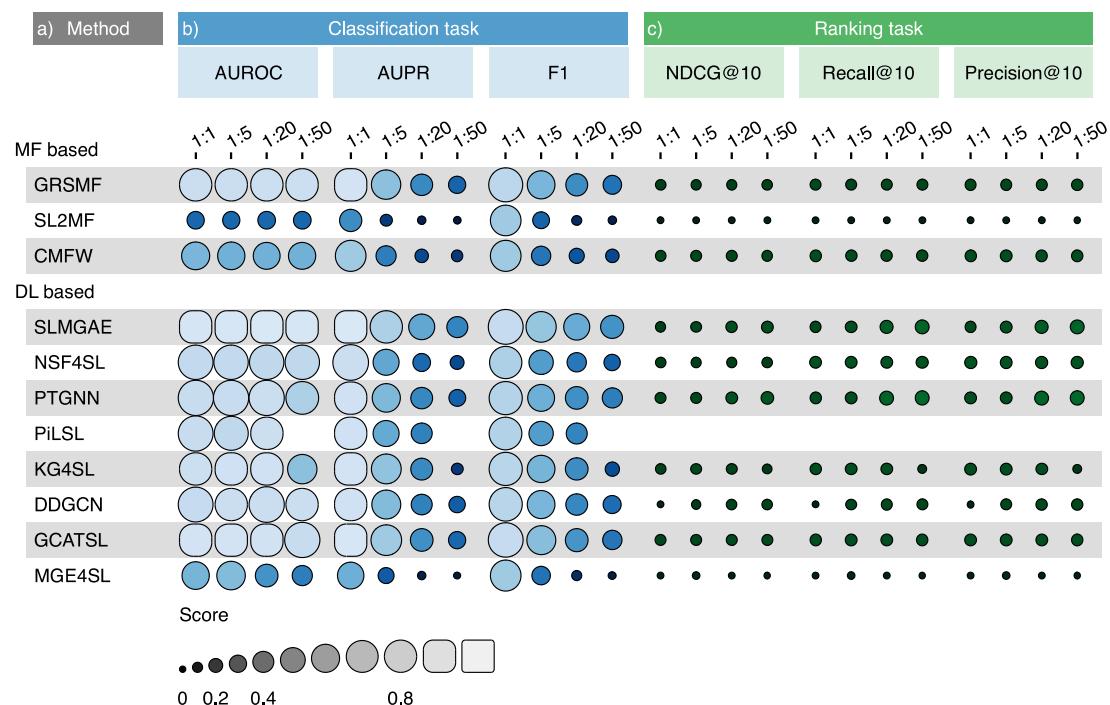


Figure S2. Result under CV2 and NSM_{Rand}

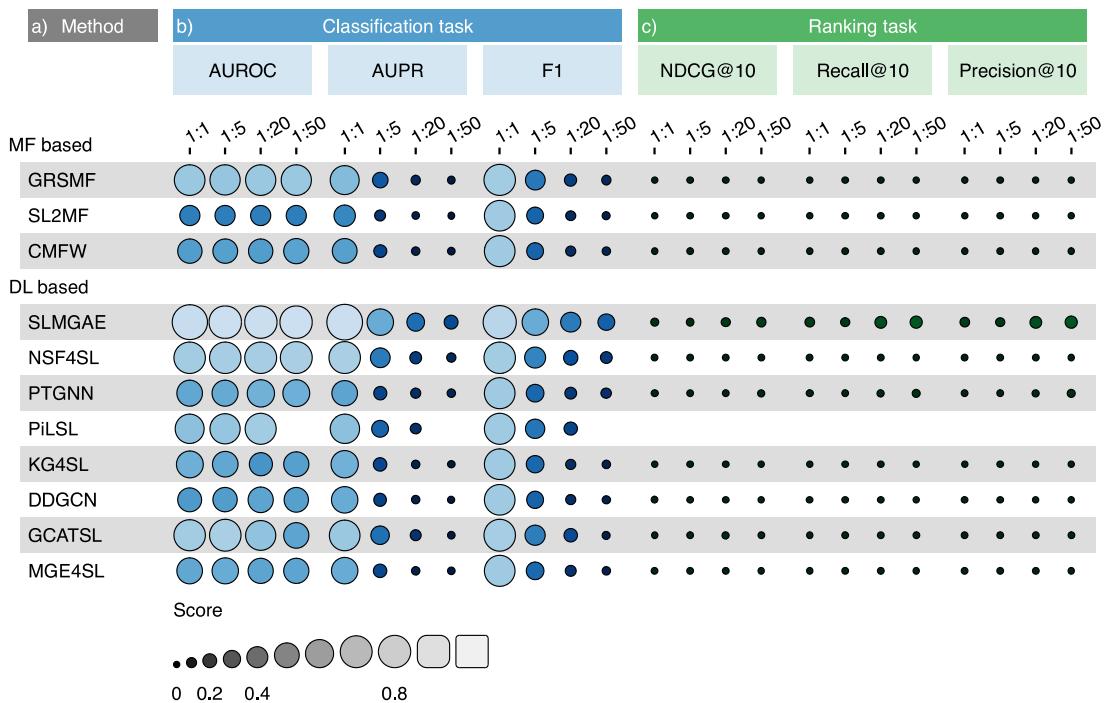


Figure S3. Result under CV3 and NSM_{Rand}

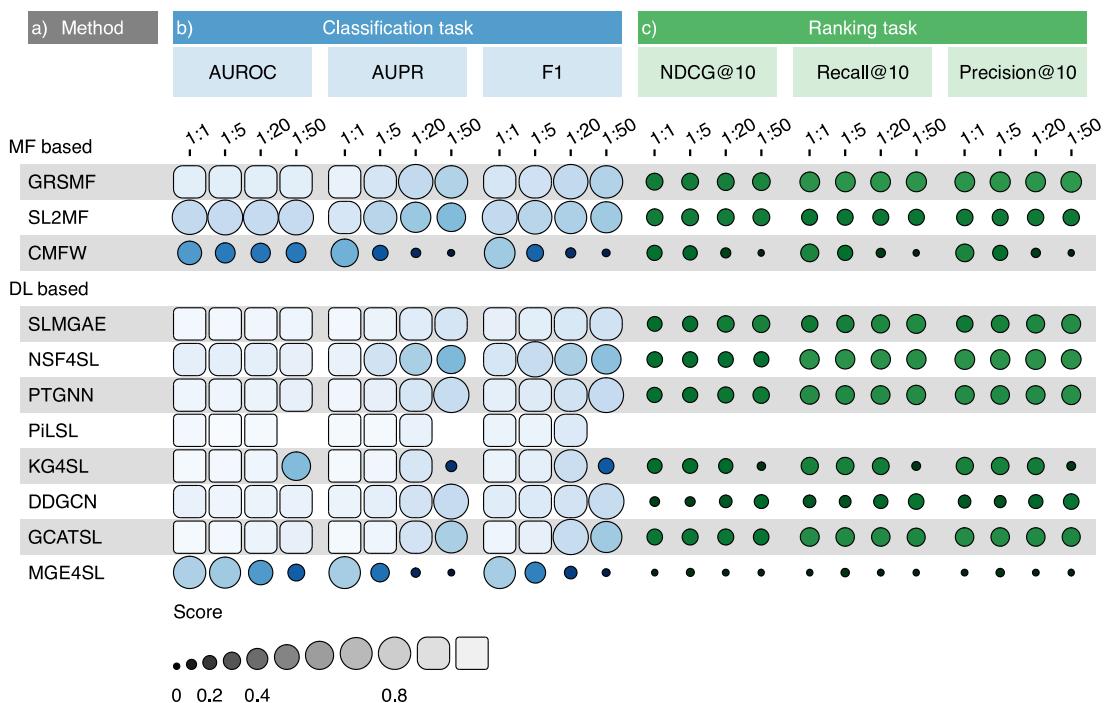


Figure S4. Result under CV1 and NSM_{Exp}

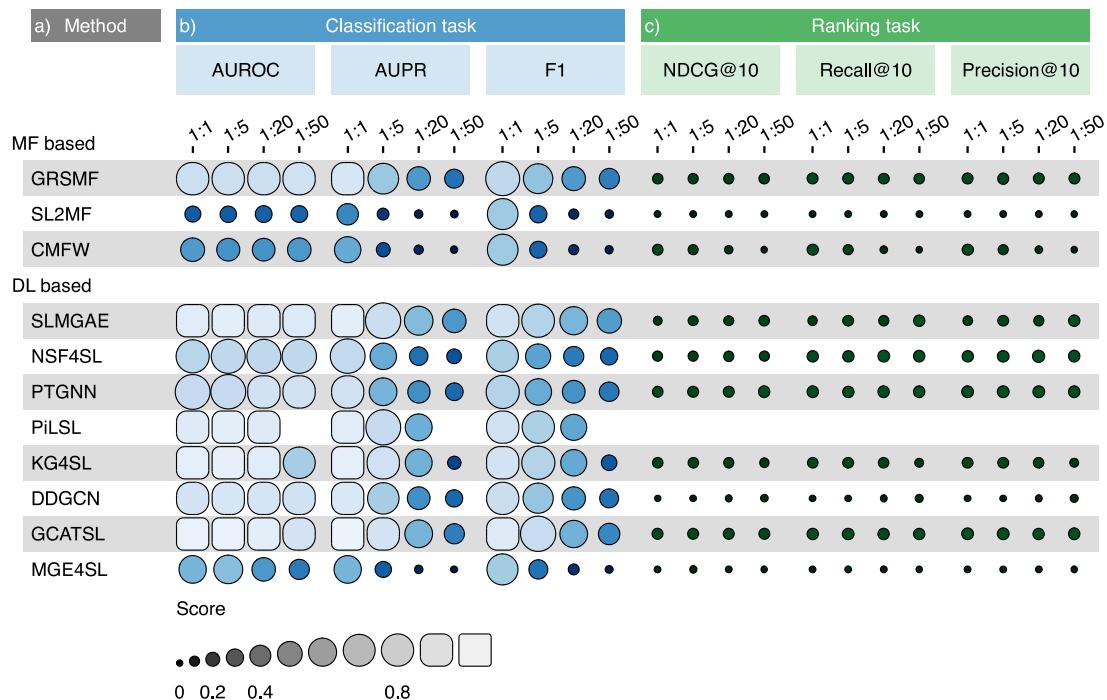


Figure S5. Result under CV2 and NSM_{Exp}

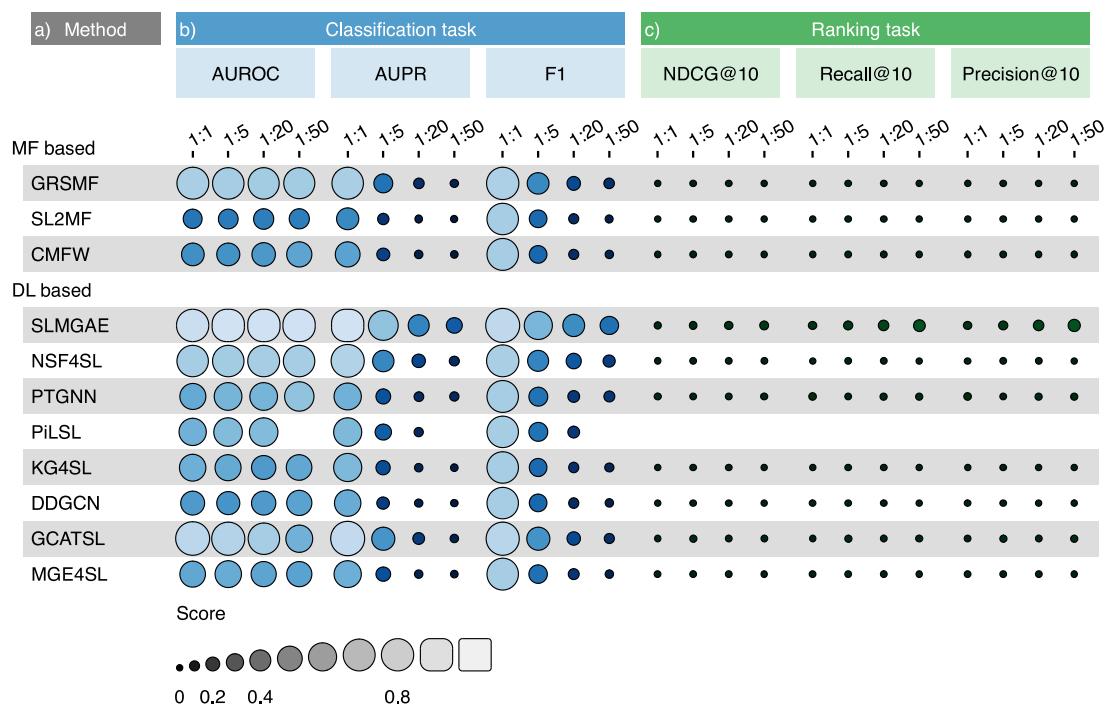


Figure S6. Result under CV3 and NSM_{Exp}

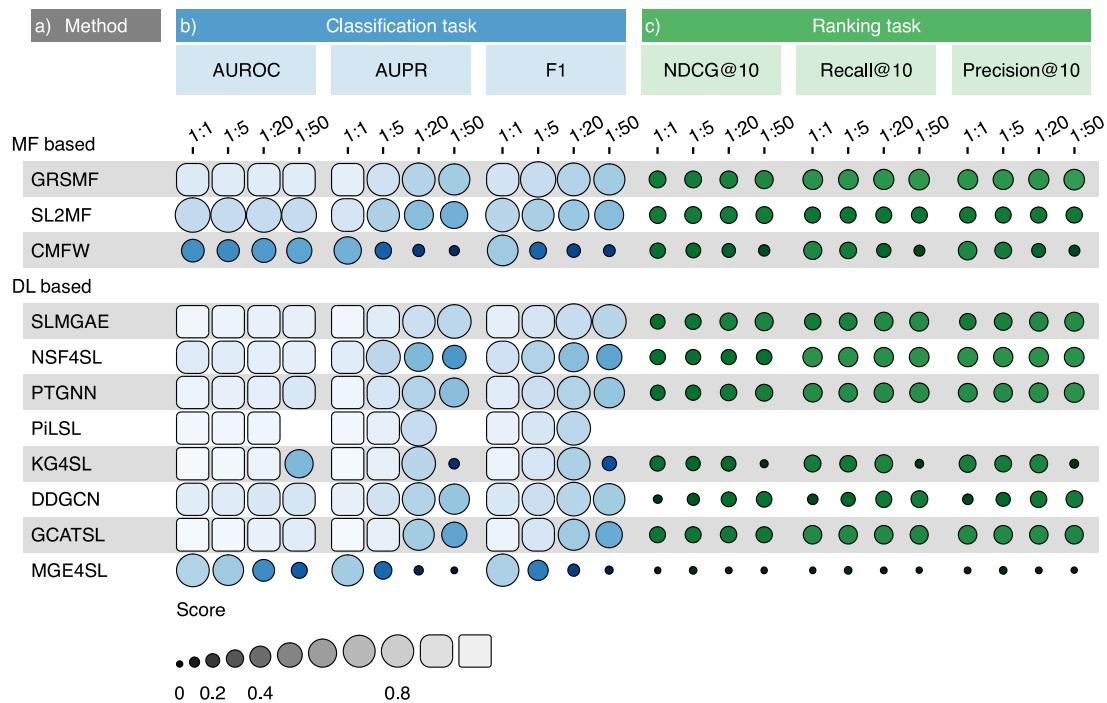


Figure S7. Result under CV1 and NSM_{Dep}

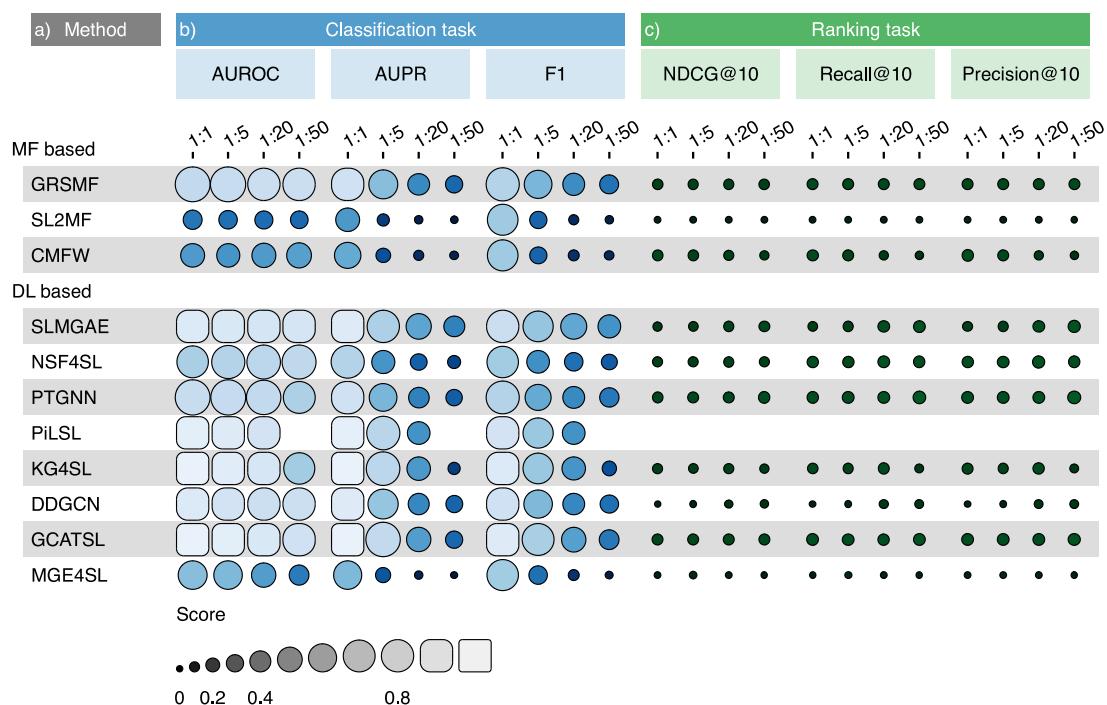


Figure S8. Result under CV2 and NSM_{Dep}

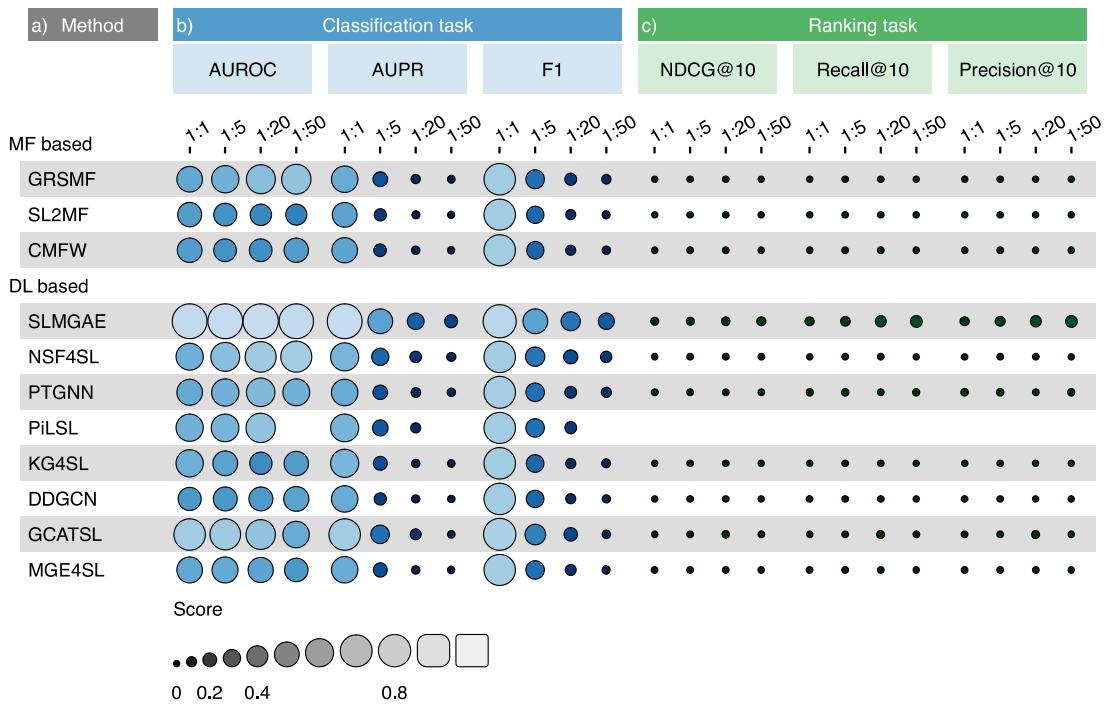


Figure S9. Result under CV3 and NSM_{Dep}

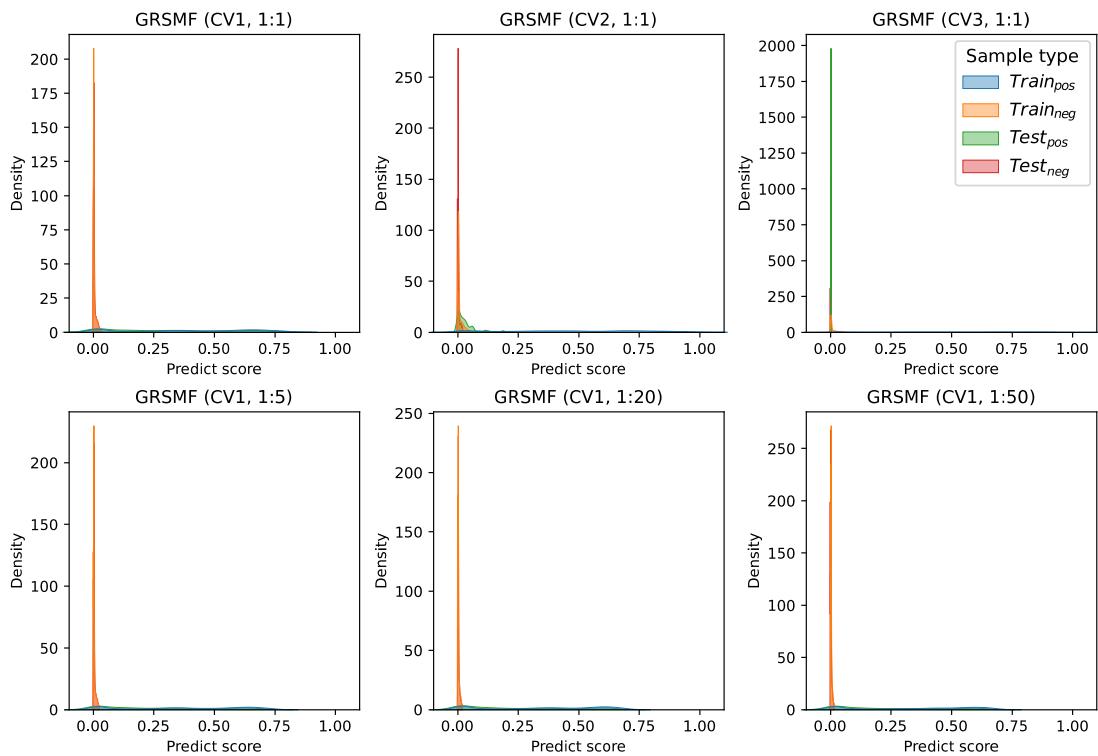


Figure S10. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the GRSMF model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.

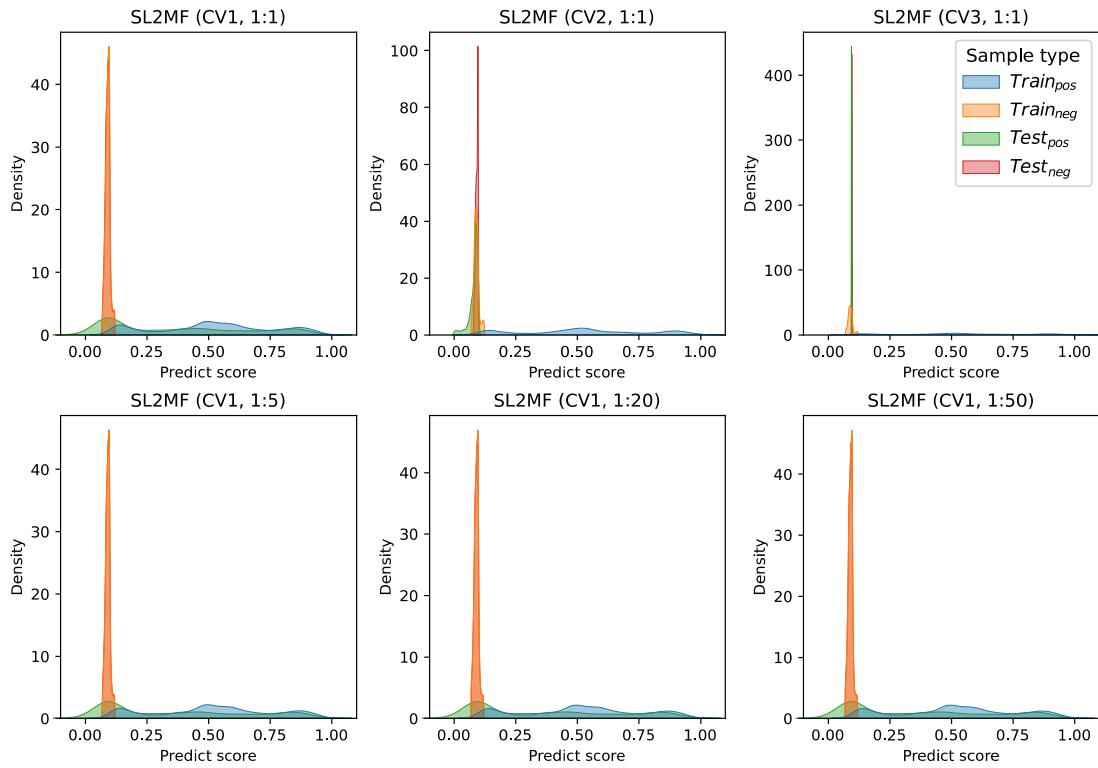


Figure S11. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the SL^2MF model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.

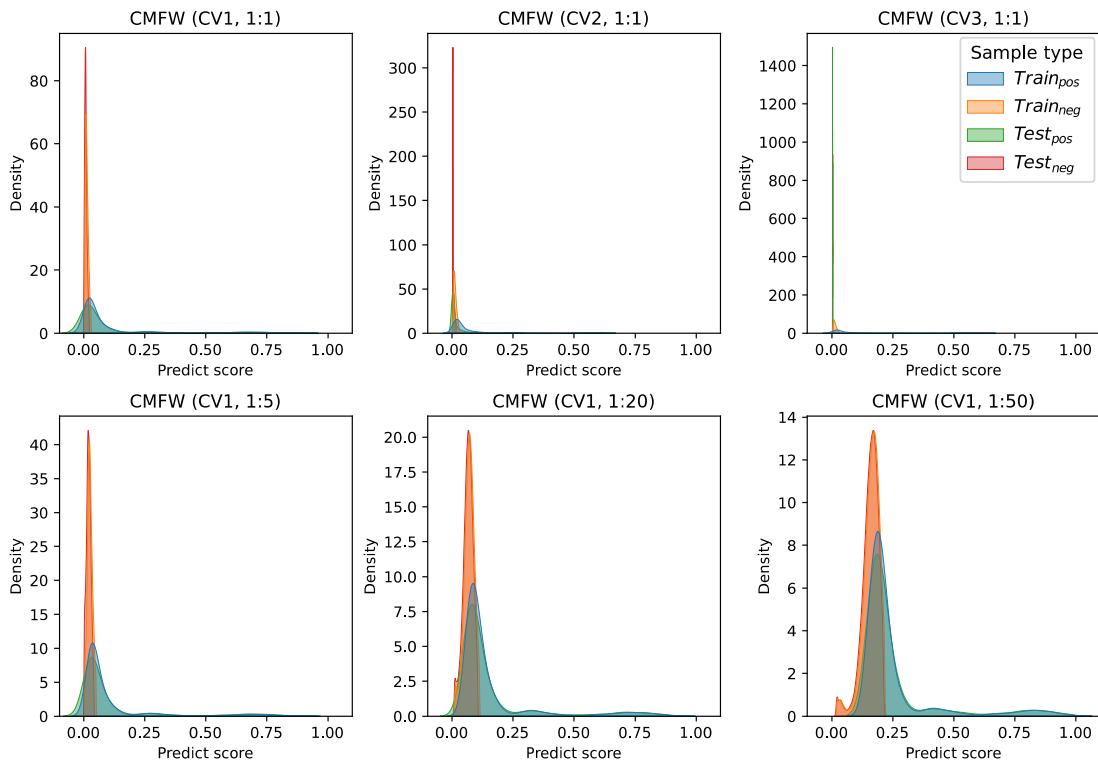


Figure S12. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the CMFW model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.

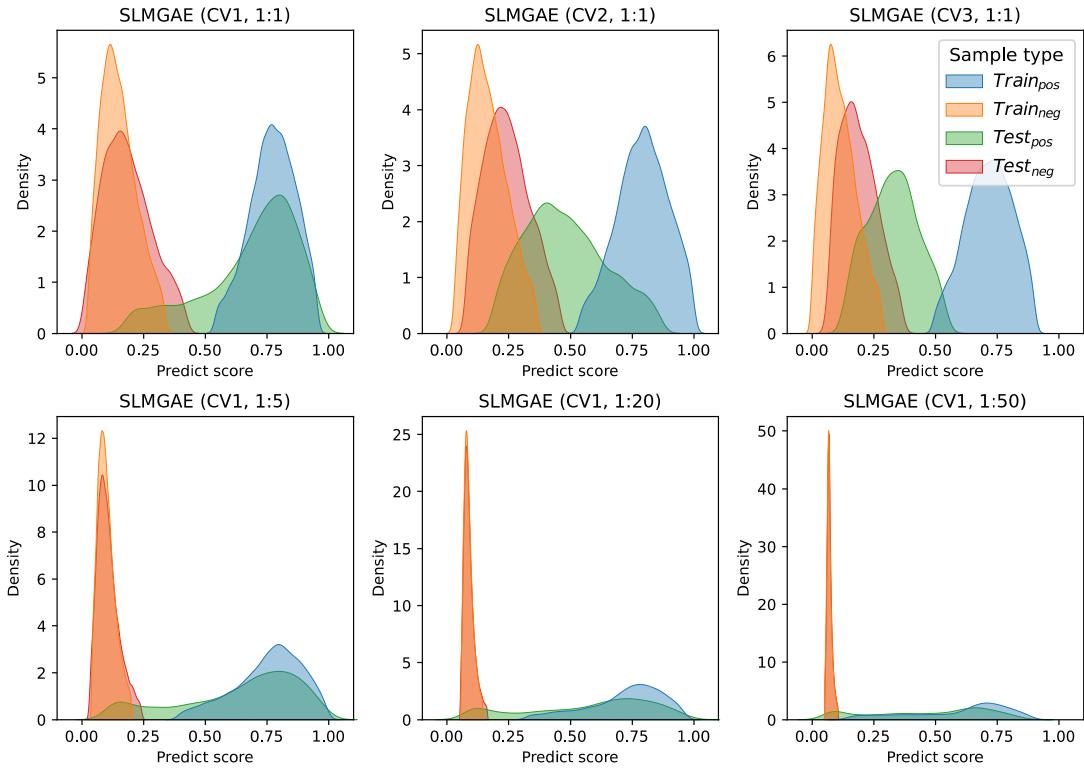


Figure S13. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the SLMGAE model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.

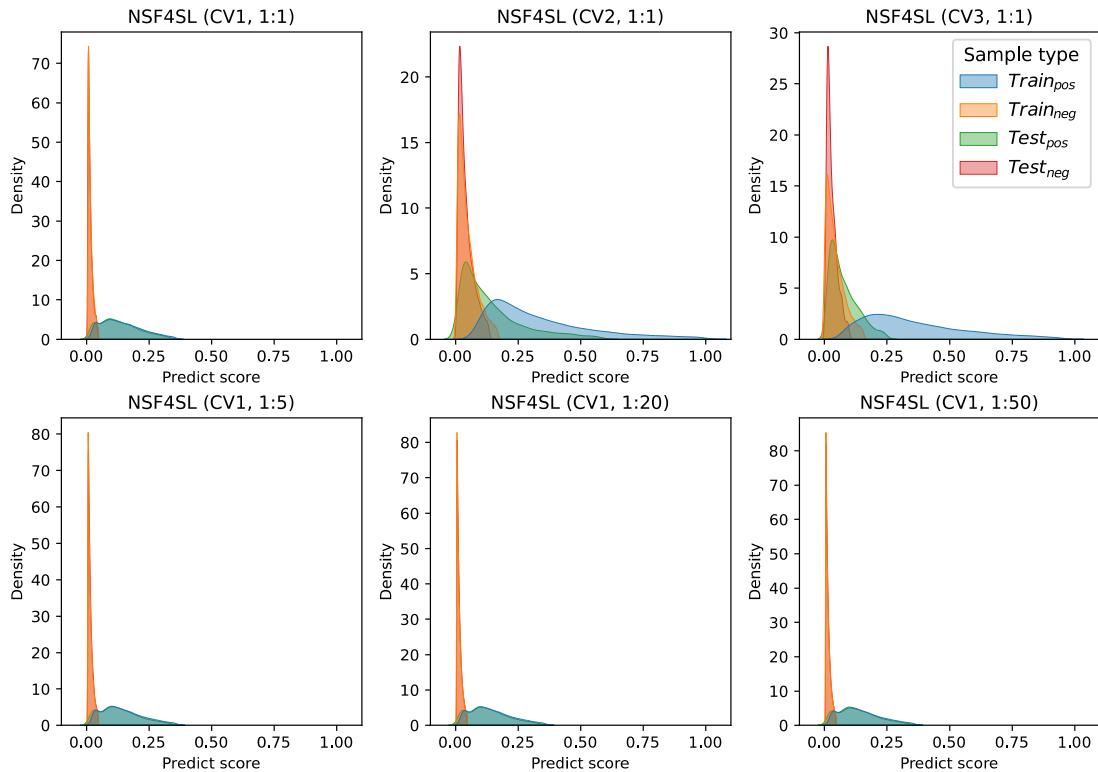


Figure S14. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the NSF4SL model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.

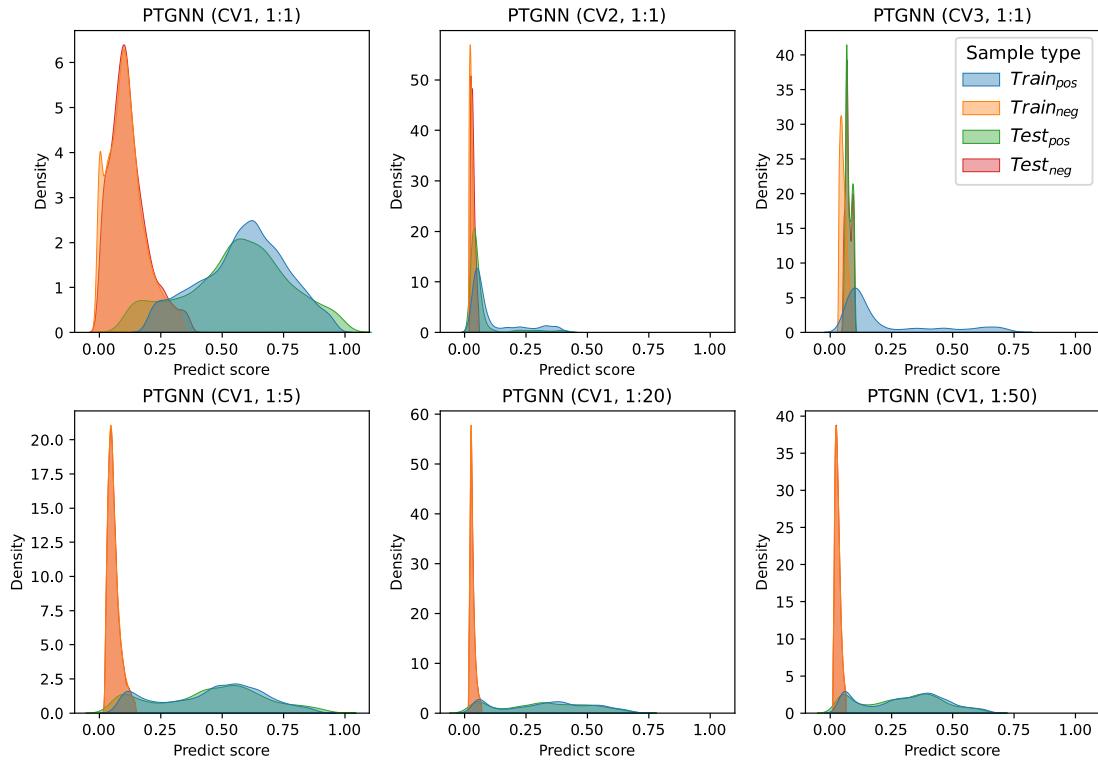


Figure S15. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the PTGNN model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.

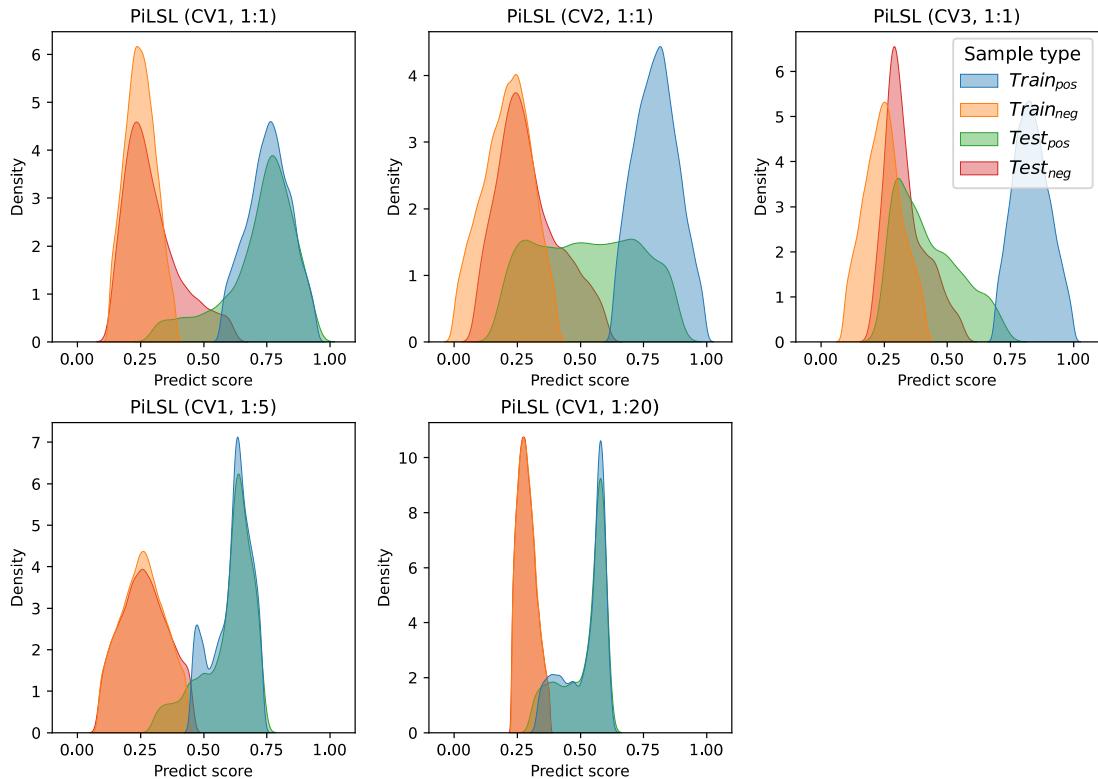


Figure S16. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the PiLSL model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.

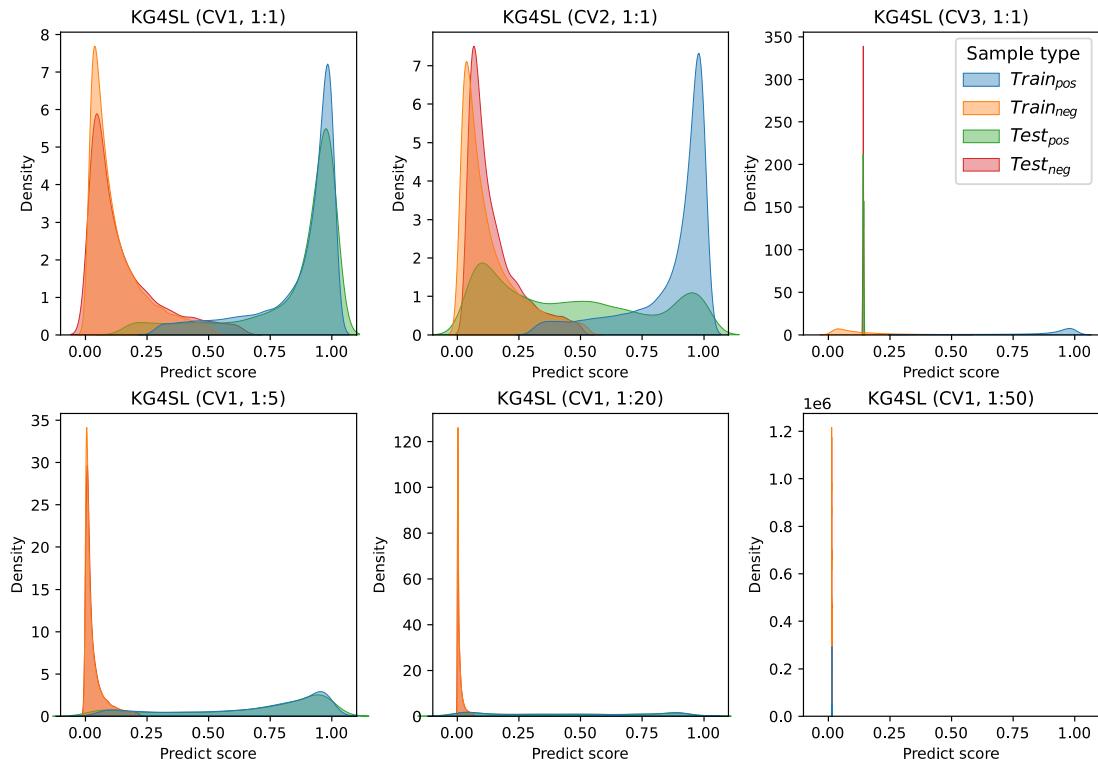


Figure S17. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the KG4SL model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.

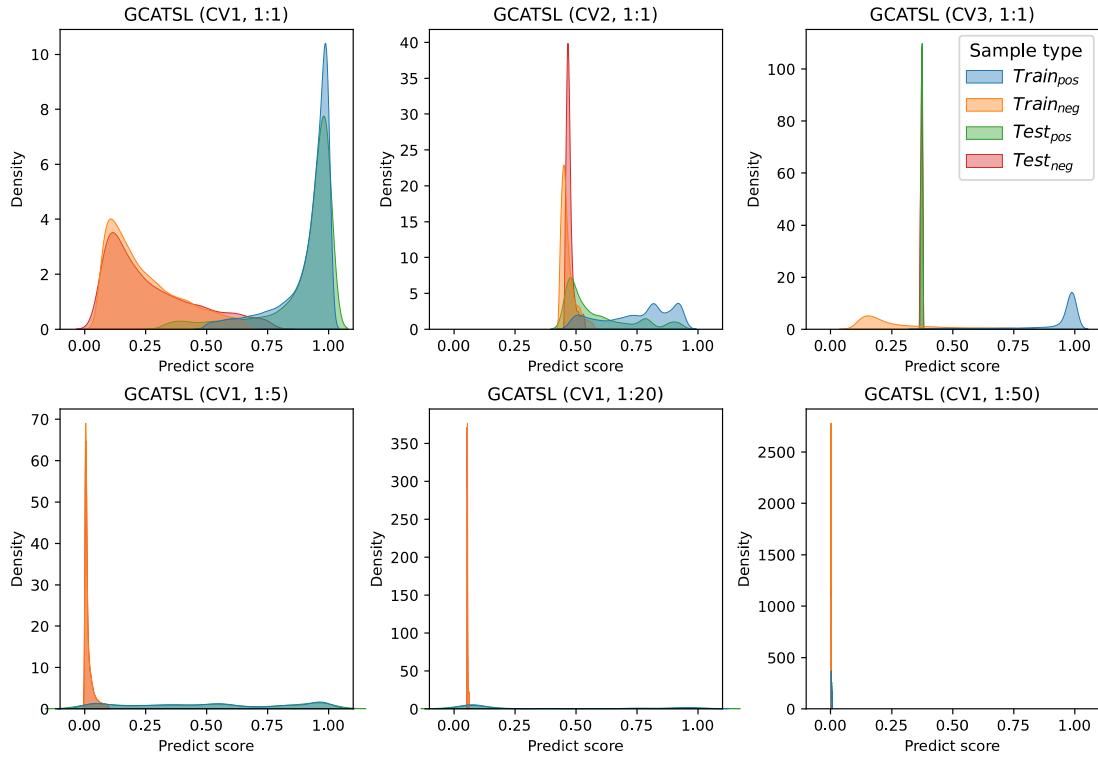


Figure S18. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the GCATSL model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.

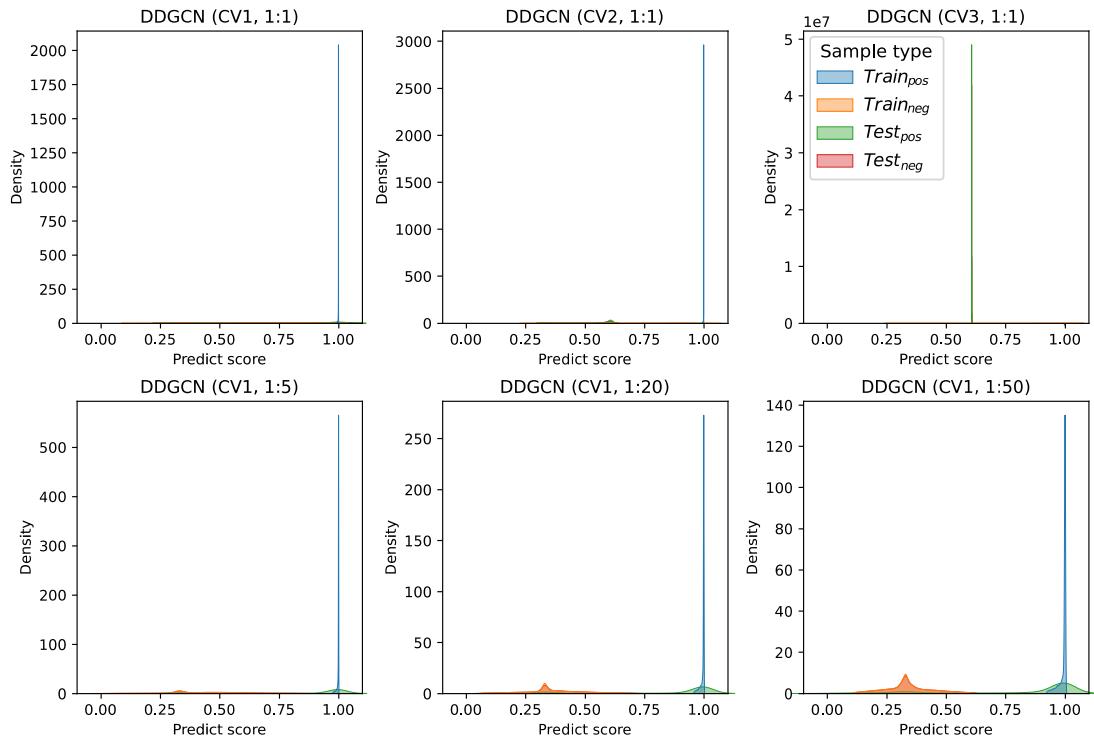


Figure S19. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the DDGCN model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.

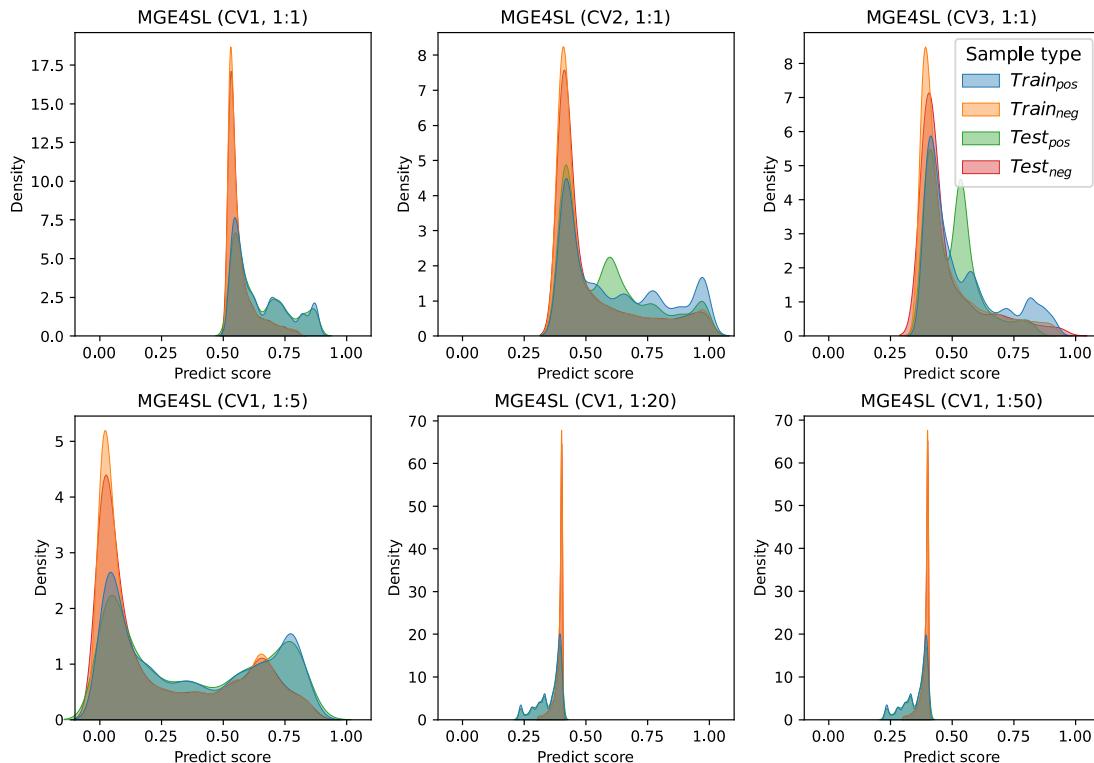


Figure S20. Distribution of predicted score. A, B, C, D, E and F show the predicted score distributions of the MGE4SL model in the training and testing sets under (CV1, 1:1), (CV2, 1:1), (CV3, 1:1), (CV1, 1:5), (CV1, 1:20), and (CV1, 1:50) scenarios, respectively.