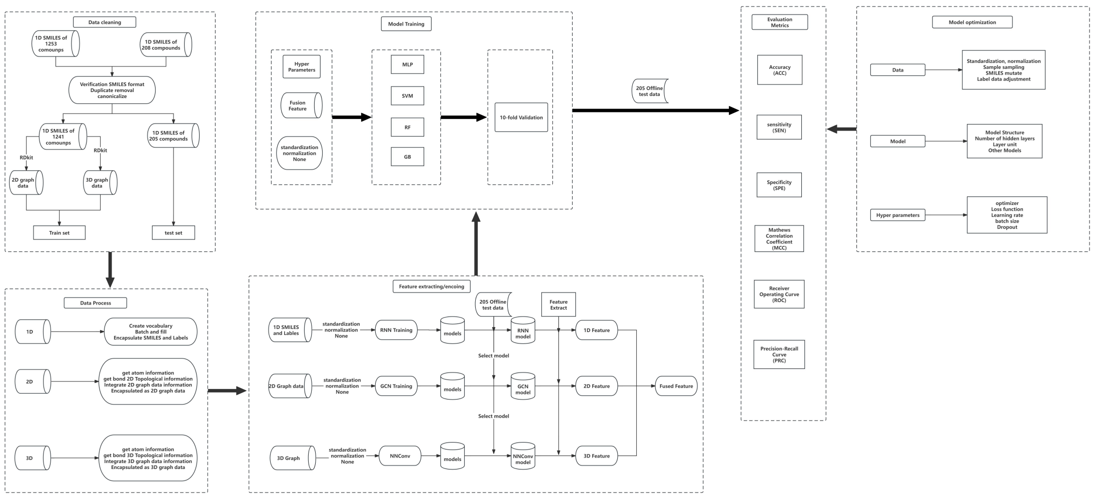
General description and high-level design of the project

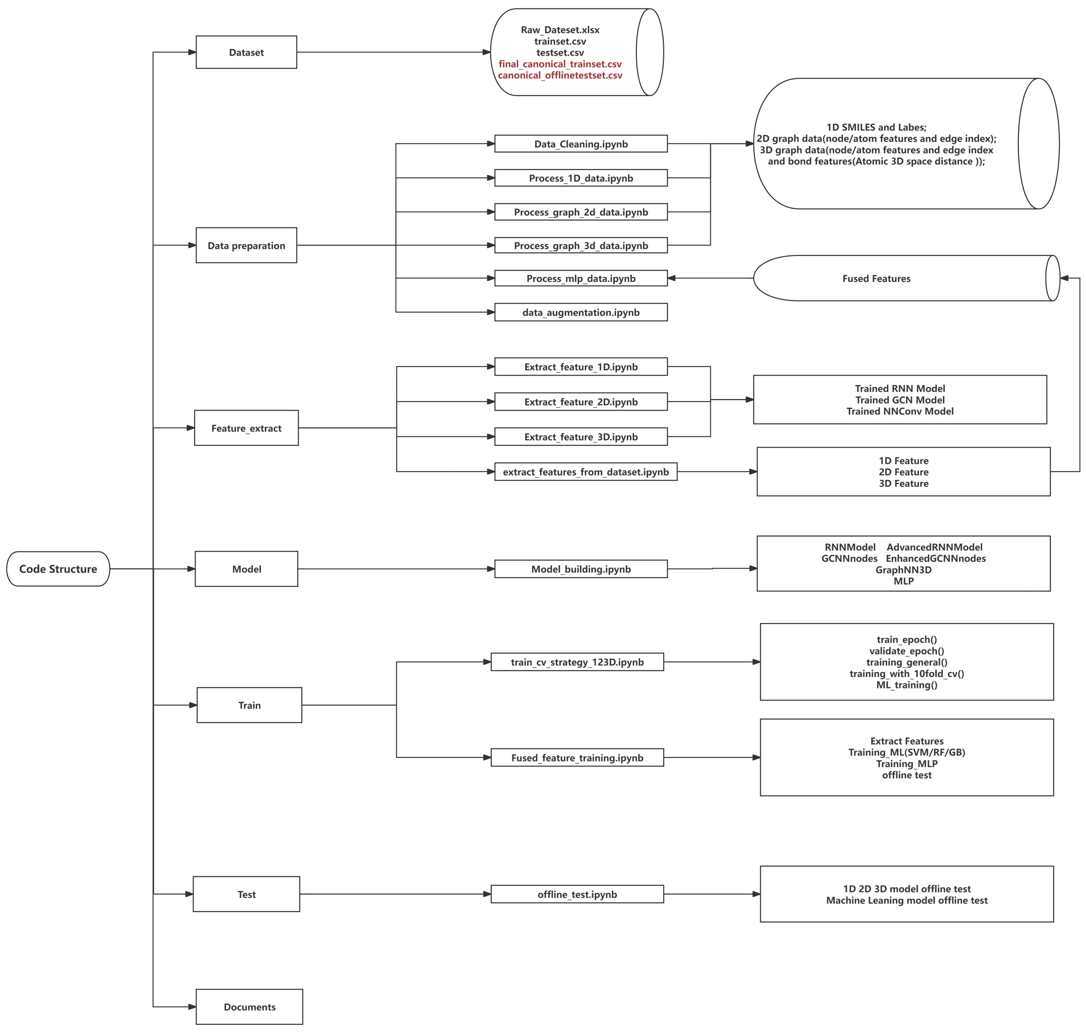
Overview:

The general idea of this project is to use different deep learning or machine learning methods to train models on multi-modal molecular data, use the trained models to extract features, and then fuse the features of different dimensions of molecular data (1D, 2D, 3D), and then Train a deep learning or machine learning model to predict the toxicity (negative and positive) of a molecule.

Project implementation can be divided into five parts, namely, data cleaning, data processing (data preparation), feature extraction and feature fusion, model training of fused features, and model tuning. The overall idea is as follows:


Data cleaning
1D SMILES of 1253 comounps
1D SMILES of 208 compounds
Verification SMILES format Duplicate removal canonicalize
1D SMILES of 1241 comounps
1D SMILES of 205 compounds
RDkit
RDkit
2D graph data
3D graph data
Train set
test set
Data Process
1D
Create vocabulary Batch and fill Encapsulate SMILES and Labels
2D
get atom information get bond 2D Topological information Integrate 2D graph data information Encapsulated as 2D graph data
3D
get atom information get bond 3D Topological information Integrate 3D graph data information Encapsulated as 3D graph data
Model Training
Hyper Parameters
Fusion Feature
standardization normalization None
MLP
SVM
RF
GB
10-fold Validation
205 Offline test data
Feature extracting/encoing
1D SMILES and Lables
2D Graph data
3D Graph
standardization normalization None
RNN Training
GCN Training
NNConv
models
205 Offline test data
Select model
Select model
RNN model
GCN model
NNConv model
Feature Extract
1D Feature
2D Feature
3D Feature
Fused Feature
Evaluation Metrics
Accuracy (ACC)
sensitivity (SEN)
Specificity (SPE)
Mathews Correlation Coefficient (MCC)
Receiver Operating Curve (ROC)
Precision-Recall Curve (PRC)
Model optimization
Data
Standardization, normalization Sample sampling SMILES mutate Label data adjustment
Model
Model Structure Number of hidden layers Layer unit Other Models
Hyper parameters
optimizer Loss function Learning rate batch size Dropout

The code implementation architecture of the project is as follows:



Part1: Data cleaning

In the public data set obtained, the SMILES type data (1D) of 1253 compounds (samples) was used as the training set, and the data of 208 compounds was used as the test set(offline test set, external data). The training set consists of 636 hepatotoxic compounds(Positive) and 617 non-hepatotoxic compounds(Negative), and the test set includes 94 hepatotoxic and 114 non-hepatotoxic compounds. Sample labels were divided into "negative" and "positive" for hepatotoxicity.

First, check the SMILES string and remove duplicates and see if these SMILES are the canonical form of the molecule. The results show that in the training set, 114 SMILES data are molecule strings in canonical form, and 1139 are molecule strings in non-canonical form, while in the test set, only 8 SMILES data are molecule strings in canonical form, and 200 are molecule strings in non-canonical form. Use RDKit tools to convert these non-canonical SMILES into canonical form.

Afterwards, check how much molecular data can generate 3D molecular conformations and can be minimally energized by the MMFF force field, set the maximum iteration number to 10000 for minimization quantization attempts, and run the code repeatedly to delete molecular data that cannot be minimized. Finally, there were 1241 molecular compounds in the training set, including 606 toxicity-negative and 635 toxicity-positive compounds, and there were 205 molecular compounds in the test set, including 111 toxicity-negative and 94 toxicity-positive compounds.

Part2: Data processing (Data preparation)

This part is divided into three sub-parts, namely, 1D SMILES data processing, 2D molecular graph data processing, and 3D molecular graph data processing. Among them, 2D and 3D molecular graph data are obtained by 1D SMILES.

Part2.1, 1D SMILES data processing

First, iterate through all SMILES strings, collect all unique characters that appear in the strings, and assign a unique index to each character. In addition, three special symbols have been added: <PAD> is used to pad shorter sequences so that all sequences are of consistent length; <SOS> represents the beginning of the sequence; <EOS> represents the end of the sequence. This vocabulary (or encoder) is what converts characters into a numerical representation that the model understands.

Then, encapsulate the SMILES string and corresponding label into a PyTorch Dataset object. For each string, it first adds <SOS> and <EOS> tags at the beginning and end of the string, respectively, and then replaces each character with the corresponding index. If a character that does not exist in the vocabulary is encountered, it is replaced with the <PAD> index. In this way, each SMILES string is converted into a sequence of integers.

Since the converted sequence lengths of different SMILES strings may be different, but the neural network requires a fixed size input, these sequences need to be padded to the same length when creating batch data. Here, a batch of sequences is padded to the length of the longest sequence in the batch, using the index of the <PAD> symbol as the padding value.

Part2.2, Processing of 2D molecular graph data

For 2D molecular graph data, first, traverses the atoms and edges in the each molecule graph to extract atomic features and edge features. For atom types, first obtain the unique atom symbols in the atom set of all molecules, with a total of 10 (plus H atoms), in preparation for constructing a vector of atomic characteristics. Atomic characteristics include atom type, atomicity, the atom's implicit valence, aromaticity, hybridization type (SP, SP2, and SP3 hybrid types are considered), and the number of hydrogen atoms. For atomic and hybridization types, the data is encoded with One-hot encoding, resulting in a fixed-length vector representation.

The final atomic feature is therefore 17 dimensions. For edge features, 2D topological information such as the conjugation of the bond, whether the bond is in a ring, and the type of the bond are extracted as edge features. This edge feature has 6 dimensions. Then use the torch_geometric library to encapsulate the edge index, node information, edge information and labels into 2D graph data.

Part2.3, Processing of 3D molecular graph data

For 3D molecular graphs, SMILES is first converted to molecular form, hydrogenated to generate 3D molecular conformations, and then quantified by MMFF field minimum energy.

Similar to the two-dimensional molecular graph, atomic information and edge information are also extracted here. But in the three-dimensional molecular graph, the spatial distance, bond angle, dihedral angle, etc. between the starting point atom and the ending point atom of each edge in each molecule are extracted as edge information. Since each edge has a different number of adjacent edges, their bond angle values are different, and the number of dihedral angles associated with the edge is different, the length of the feature of each edge may be different, so the bond angle and dihedral angle of the edge take the maximum, minimum, and average values of their values as the information of bond angles and dihedral angles.

Finally, the dimension of the information of each edge is 7 dimensions. Voxelized data were also generated using three-dimensional coordinates. Finally, these information is encapsulated into 3D graph data.

Part3: Feature extraction and feature fusion

This part is to perform deep learning modeling on the processed 1D, 2D, and 3D molecular data, train the model, and use the well-performing model to extract the features of these three types of molecular data. This part can be divided into 4 sub-parts, namely model construction, model training, model selection, feature extraction and fusion.

Part3.1, model construction

Here, RNN is used to model 1D SMILES; for 2D molecular graph data, GCN is used to model node information and edge index. The 2D topological information of the edge is not used here, so the input data is 17 dimensions of atomic features; for 3D molecular graphs, only the distance between atoms on the edge (1 dimension) is used as the edge information,then the edge information and the atomic information (17 dimensions) and edge index are modeled using NNConv.

RNN uses a simple LSTM network, which is divided into embedding layer, LSTM layer and fully connected layer.

The embedding layer maps the input sequence (integer index) to a fixed-size embedding vector. input_dim is the vocabulary size (i.e. the number of unique tokens the model can handle) and embed_dim is the dimension of the embedding vector. The role of this layer is to transform discrete, sparse input into a denser, information-rich representation that can be better processed by subsequent layers.

The LSTM layer uses long short-term memory (LSTM) units to process sequence data. embed_dim is the input dimension of the LSTM layer (the same as the output dimension of the embedding layer), and hidden_dim is the dimension of the LSTM hidden state. batch_first=True means that the first dimension of the input and output tensors is the batch size.

The fully connected layer maps the output (hidden state) of the LSTM layer to the output space. hidden_dim is the input dimension of the fully connected layer (the same as the hidden state dimension of LSTM), and output_dim is the output dimension of the model.

In forward propagation, first, the text sequence (represented by index) is fed into the embedding layer to obtain the embedding representation of each token.
The embedded sequence is then passed to the LSTM layer, which processes the entire sequence and outputs the output of each time step along with the final hidden and cell states (not used in this model).

Finally, the hidden state of the last time step is compressed (to remove the batch dimension, if the batch size is 1) and transformed into the final output dimension through a fully connected layer.

The GCN model contains graph convolution layers and fully connected layers. The GCNConv layer is used to process the nodes and edges of the graph. These layers update the feature representation of each node by aggregating information from neighbor nodes and can capture the local structural characteristics of the graph. num_node_features is the number of features for each node in the graph (i.e. input dimensions).

In forward propagation, the input data contains three key information, namely, x (node feature matrix), edge_index (edge index, describing the structure of the graph), and batch (the index of the graph to which each node belongs in the batch , used for pooling operations), during the 2D molecular graph modeling process, no edge information is added.

The input node features are first processed through two GCNConv layers, and the ReLU activation function is applied after each layer. In this process, the features of each node are updated based on the features of its neighbor nodes. After the node updates the features through the graph convolution layer, global average pooling is used to aggregate all node features of each graph into a graph-level representation.

The graph-level representation is further processed through a fully connected layer to produce the final output.

The NNConv model enhances graph representation by combining node features and edge features.
Two graph convolutional layers are used here to process node features. These layers help capture the topology of the graph by exploiting the connection patterns between nodes to update the feature representation of nodes. The model is designed with edge features (edge_attr) in mind, which can represent attributes of edges or relationships between pairs of nodes. The model combines node and edge information by performing a simple mean aggregation of edge features and splicing it with node features obtained through global pooling.

In forward propagation, the input data includes node features (x), edge index (edge_index), edge features (edge_attr) and batch information (batch).
The input node features first pass through two graph convolutional layers, and the ReLU activation function is applied. Aggregate features from all nodes in the graph into a single graph representation using global average pooling. Edge features are aggregated by means and spliced with graph-level features of nodes.
Finally, the concatenated graph representation is further processed through two fully connected layers to generate the final output.

In addition, multiple different models were built to compare model tuning and training effects, mainly by designing different layers and changing the input and output of feature dimensions between layers, as well as adding dropout layers or Batch normalized layers to the model.

Part3.2, model training

During model training, the data is first preprocessed, including normalization, standardization, and no processing. These processes are optional. During preprocessing, the preprocessing file is saved for subsequent model selection(test) and used in the testing phase to ensure the consistency of the model on the training set and test set. In addition, data augmentation methods are also set up, that is, copying the original molecular data or generating different forms of SMILES molecules for the molecules.

In the training phase, there are two loss functions available, BCEWithLogitsLoss and CrossEntropyLoss, which are 1-dimensional and 2-dimensional for the model output dimensions respectively. BCEWithLogitsLoss is used by default here, and the optimizer uses the Adam algorithm here. During training, the data is divided into training set and validation set, the proportion of which is 0.8 and 0.2 respectively.

Finally, ACC, SEN, SPE, and MCC are used to measure the model's score on the validation set. The learning rate adopts a fixed learning rate, such as 0.01, 0.001, etc.

In addition to conventional training, 10 fold cv training was also performed, and the parameters of the model were not reset for each fold of data. This was designed to learn more knowledge for subsequent feature extraction.
When the model training is completed, adopt certain strategies to save the model, such as saving the best model, or saving a model every 5 or 10 epochs.

Part3.3, model selection

After the model is trained, the saved models are tested on the offline test set to select a good model for later feature extraction. During testing, data processing adopts the same process as during training, such as standardization.

Part3.4, feature extraction and fusion

After offline testing, select a model that performs well on both the validation set during training and the offline test set to avoid overfitting or underfitting, and then use the model to extract data. In this process, can also Perform data augmentation on data, such as copying multiple copies of data or generating molecular variants.

Part4: Model training of fused features

After extracting features from data of different dimensions, use machine learning or deep learning models to train the fused feature data.
Here, support vector machines, random forests, Gradient Boosting Trees and multi-layer perceptrons are used to train features respectively. During training, 10-fold cross-validation is used, and pre-processing operations are also set for the data, such as standardization, normalization, PCA dimensionality reduction. Finally, the trained model is tested on the offline test set.

Judging from the current results, using random forest for 10 times of cross-validation training has the best effect. The average results are: ACC: 0.9220, Average SEN: 0.9312, Average SPE: 0.9123, Average MCC: 0.8440.
On the offline test set, the performance is ACC: 0.7707, SEN: 0.9255, SPE: 0.6396, MCC: 0.5800.

Part5： model tuning

It is obvious that the performance of the final model currently trained on the fused feature set and the test set shows that the model is overfitting.

At the same time, some problems also occurred during the training process of the feature extraction model. Regarding the performance of the RNN model on 1D data, the learning rate dropped rapidly during the training process, but the model effect did not improve accordingly. During the training process of 2D and 3D molecular graph data, the learning rate decreases slowly and the effect also improves slowly. However, as the epoch increases, the learning rate and effect improve, but in the offline test data set (used for the model selecting), the effect is not good, which is also the occurrence of over-fitting.

During the model tuning process, different preprocessing, epoch, batch size, learning rate, data augmentation methods, and different model structures are used to conduct comparative training on the data. The results show that after the feature extraction models are optimized, the training effect of the fused features on the machine learning model is improved. In other words, only if the feature extraction models are good, the extracted features will be more conducive to the training of the machine learning algorithm. However, the robustness of the models need to be improved.

The next plan is to print out the detailed information of the test during feature extraction model training and machine learning model training, such as viewing the confusion matrix to see the model's performance on different labels, and then conduct more detailed data augmentation and other methods to perform model tuning.