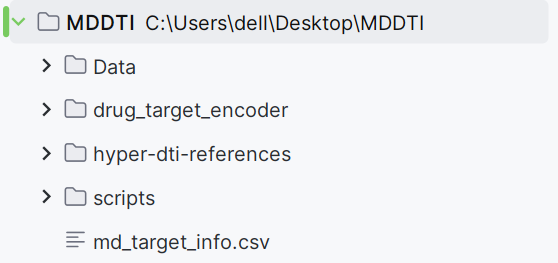
**精神类疾病的DTI预测模型说明文档**

2020200730 姚玉龙

# 项目架构



Data: 项目使用的数据集和最终产生的结果；

drug\_target\_encoder: 计算药物分子和蛋白靶标结构的表征方式；

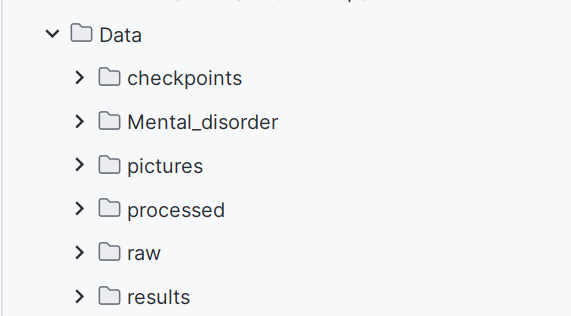
hyper-dti-references: 本次研究参考的文献代码；

[ml-jku/hyper-dti: HyperPCM: Robust task-conditioned modeling of drug-target interactions (github.com)](https://github.com/ml-jku/hyper-dti)

scripts: 完成DTI预测的机器学习代码；

md\_target\_info.csv: UniProt完成检索的靶标信息，和source.csv一致，提到外面方便查阅；

# 数据部分（Data）



checkpoints: 计算MolBert分子表征使用的模型权重文件------last.ckpt

CDDD分子表征方法的文件也下载完成，但具体实验过程中因为tensorflow模型加载权重文件出错（可能是环境配置，具体还未解决），选择放弃；

Mental\_disorder: 完成精神类疾病的筛选工作，具体数据集都在该文件夹下；

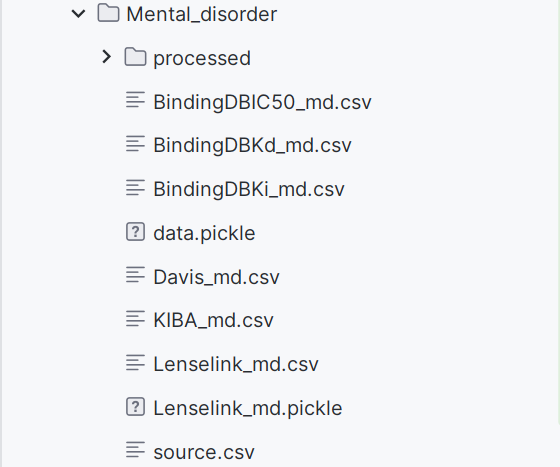
pictures: 此次研究中绘制的所有图片位置；

processed: 对于未筛选的数据集实验的部分表征文件；

raw: 未进行筛选的原始数据集（从TDC平台获取）

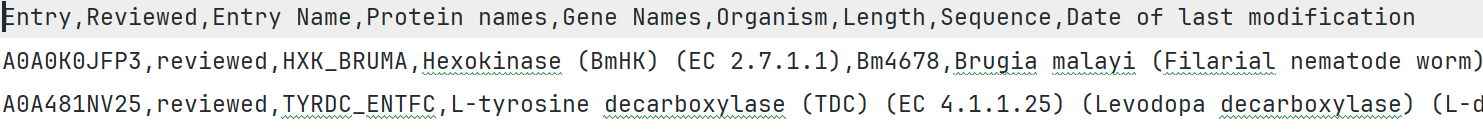
results: 代码后运行保存的所有结果；

## 精神类疾病Mental\_disorder:



data.pickle: HyperPCM文献中的Lenselink数据集文件；

**source.csv: 筛选整合的精神类疾病药物信息，共1889条；**

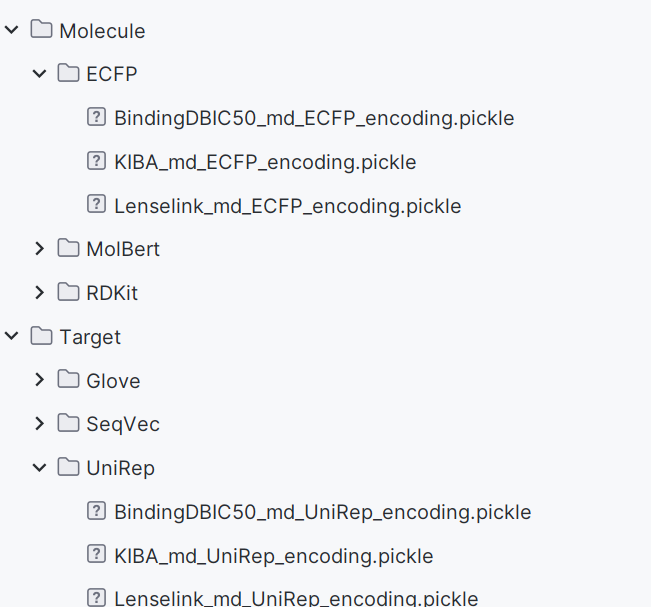


其中，可通过**Entry**在其他蛋白靶标的数据库中获取三维空间信息；

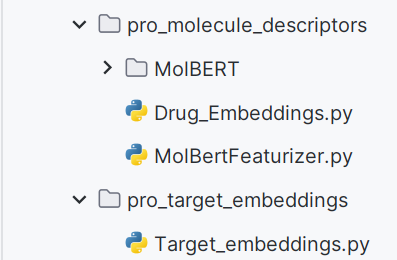
**\_md**: 表示数据集 均已完成筛选；

补充：Lenselink数据集的pickle文件可能因为pandas库版本冲突，造成读写过程中报错，最终选择csv文件；

processed: 项目使用的编码文件；



# 药物分子和靶标结构的编码代码（drug\_target\_encoder）



## pro\_molecule\_descriptors

注：创建MolBert库的conda环境，计算分子表征方式；[BenevolentAI/MolBERT (github.com)](https://github.com/BenevolentAI/MolBERT)

Drug\_Embeddings.py: 计算ECFP和RDKit；

MolBertFeaturizer.py: 计算MolBert；

## pro\_target\_embeddings

注：创建Bio-embeddings环境，同时观察SeqVec和UniRep使用的库版本是否冲突；

[sacdallago/bio\_embeddings: Get protein embeddings from protein sequences (github.com)](https://github.com/sacdallago/bio_embeddings)

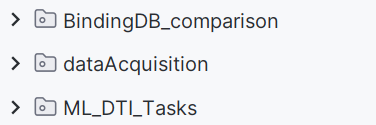
SeqVec选择CPU-only版本，GPU版本直接报错，暂时未完成debug；

UniRep若使用GPU版本的需要调整CUDA的版本问题；但此实验不必训练，直接使用模型即可，要求不高可直接使用CPU版本；

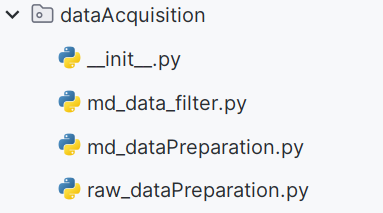
# 代码参考(hyper-dti-references)

主要参考hyper-dti-references/hyper\_dti/baselines/tabular\_baselines.py文件完成数据集预处理和DTI预测任务；

# 机器学习代码（scripts）



## dataAcquisition

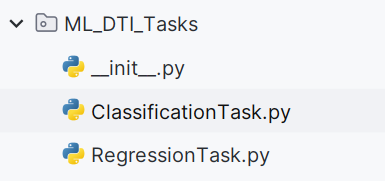


Filter：筛选KIBA、Davis、BindingDB和Lenselink数据集的代码，利用isin()方法；

md\_dataPreparation: 完成药物分子和靶标信息的编码处理，返回训练集和测试集，功能类似于train\_test\_split;

raw\_dataPreparation: 原始数据集的信息编码处理，基本属于实验尝试阶段；

## ML\_DTI\_Tasks



分类任务和回归任务处理代码架构基本一致：

* 模型的初始化和参数设置；
* 定义计算模型性能分数和图（分类-ROC和PR曲线，回归-预测值和真实值分布关系）的代码；
* 针对每个机器学习模型单次训练的代码，run\_single();
* 定义多次训练最终保存结果的代码，run\_mutiple();

其中，分类任务可完成超参数优化，但由于后期时间紧张，最终选择放弃执行优化代码。

## BindingDB\_comparision

对最终模型性能的汇总展示，处理之前保存的数据，使用jupyter书写，无太多需要注意的内容。