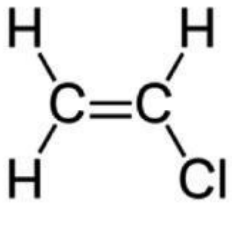
NeurIPS- Open Polymer Prediction 2025

赛后理解

竞赛任务介绍：多目标回归任务

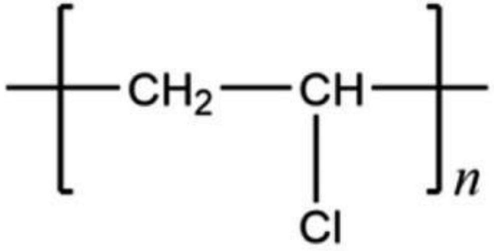
- 任务：从聚合物 SMILES 直接预测五项物性（Tg / FFV / Tc / Density / Rg）
- 意义：支撑可持续材料的虚拟筛选与加速发现，减少昂贵的实验与仿真迭代

(a).



C=CCl

(b).



[*]CC(Cl)[*]

```
data
```

	id	SMILES
0	87817	*CC(*)c1ccccc1C(=O)OCCCCCC
1	106919	*Nc1ccc([C@H](CCC)c2ccc(C3(c4ccc([C@@H](CCC)c5...
2	388772	*Oc1ccc(S(=O)(=O)c2ccc(Oc3ccc(C4(c5ccc(Oc6ccc(...
3	519416	*Nc1ccc(-c2c(-c3ccc(C)cc3)c(-c3ccc(C)cc3)c(N*)...
4	539187	*Oc1ccc(OC(=O)c2cc(OCCCCCCCCCOCC3CCCN3c3ccc([N...
...	...	...
7968	2146592435	*Oc1cc(CCCCCCCC)cc(OC(=O)c2cccc(C(*)=O)c2)c1
7969	2146810552	*C(=O)OCCN(CCOC(=O)c1ccc2c(c1)C(=O)N(c1cccc(N3...
7970	2147191531	*c1cc(C(=O)NCCCCCCCC)cc(N2C(=O)c3ccc(-c4ccc5c(...
7971	2147435020	*C=C(*)c1ccccc1C
7972	2147438299	*c1ccc(OCCCCCCCCCCCOC(=O)CCCCC(=O)OCCCCCCCCCCC...

7973 rows × 7 columns

竞赛数据与指标

- 标签源自多次**分子动力学模拟**的均值；隐藏测试集规模约 1.5K。
- 指标：wMAE（按样本稀缺度与取值范围重加权，确保各任务同等重要）。
- 代码限制：Kaggle Notebook、禁网、单次 CPU/GPU 运行 ≤ 9 小时。

The evaluation metric for this contest is a **weighted Mean Absolute Error (wMAE)** across five polymer properties, defined as:

$$\mathrm{wMAE} = \frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} \sum_{i \in \mathcal{I}(X)} w_i \cdot |\hat{y}_i(X) - y_i(X)|$$

where $\mathcal{X}$ is the set of polymers being evaluated, and $\mathcal{I}(X)$ is the set of property types for a polymer X. The term $\hat{y}_i(X)$ is the predicted value and $y_i(X)$ is the true value of the i-th property.

To ensure that all property types contribute equally regardless of their scale or frequency, we apply a **reweighting factor** w_i to each property type:

$$w_i = \left(\frac{1}{r_i}\right) \cdot \left(\frac{K \cdot \sqrt{1/n_i}}{\sum_{j=1}^{K} \sqrt{1/n_j}}\right)$$

where n_i is the number of available values for the i-th property, and $r_i = \max(\mathcal{Y}_i) - \min(\mathcal{Y}_i)$ is the estimated value range of that property type based on the test data. K is the total number of tasks. This design has three goals:

算法基本流程

1. 数据治理与外部监督扩充

过滤聚合物记号中的 R-group 与可疑模式，使用 RDKit 进行可解析性校验并统一转换为 canonical SMILES，从输入层面保证同构分子的一致表示。在此基础上，加入多源 Tg/Tc/Density/FFV 外部数据，以 canonical SMILES 为键进行分组与均值聚合以去重。

2. 多视角表征、特征工程与模型训练

a) 分子图表征（**GNN 路线**）：将分子解析为节点与边的图结构：节点特征包含原子序数、原子度、形式电荷、杂化态、芳香性、总氢数、是否位于环以及原子质量；边特征包含键类型、是否在环、是否共轭以及是否芳香，并以无向方式构图；此外补充分子级全局特征（MolWt、HBD、HBA、TPSA、可旋转键数与 SMILES 长

度)，以强化图级表达。模型以 GCN 堆叠提取局部结构，再接入 GAT 注意力层以强化信息聚合；随后对节点表示进行 mean 与 max 池化并与分子级特征拼接，在输出端为每个目标配置独立的回归头。

b) 化学描述符表征（**CatBoost 路线**)：使用 Mordred 计算大规模 2D 描述符，训练侧基于预计算特征表并去除常数列与非数值列，测试侧对 test 集进行在线计算并与训练集列对齐。基于 Mordred 描述符，使用 CatBoost 模型进行回归。

c) 指纹与图统计表征（**XGBoost 路线**)：构建 Morgan 指纹（半径 2128 位）与 MACCS 166 位指纹，并提取 RDKit 物化描述符以及基于 NetworkX 的图统计量（例如图直径、平均最短路径与环数量)。最终输入 XGBoost 模型进行训练。

3. 模型融合

采用线性加权进行模型融合，加权方案为 GNN 占比 0.4，CatBoost 占比 0.3，XGBoost 占比 0.3，五个目标分别独立加权。

算法流程图

数据治理

SMILES 清洗/标准化外部数据并入三视角

建模

GNN | Mordred+CatBoost | 指纹

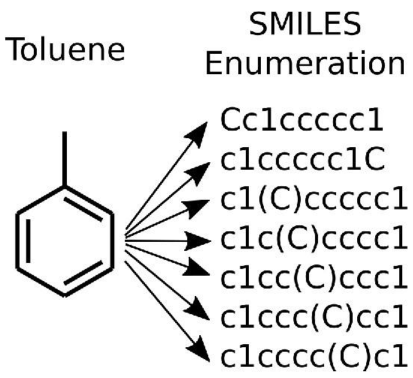
+图统**+XGB**

加权融合

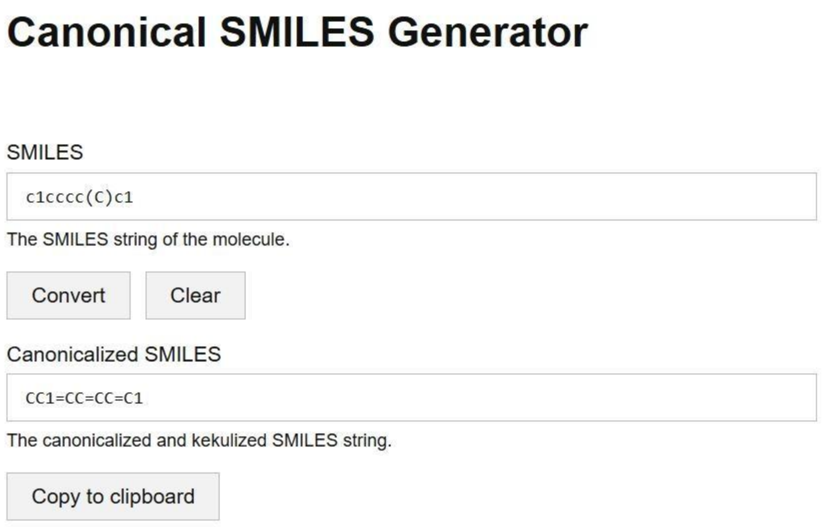
GNN 0.4 / Cat 0.3 / XGB 0.3

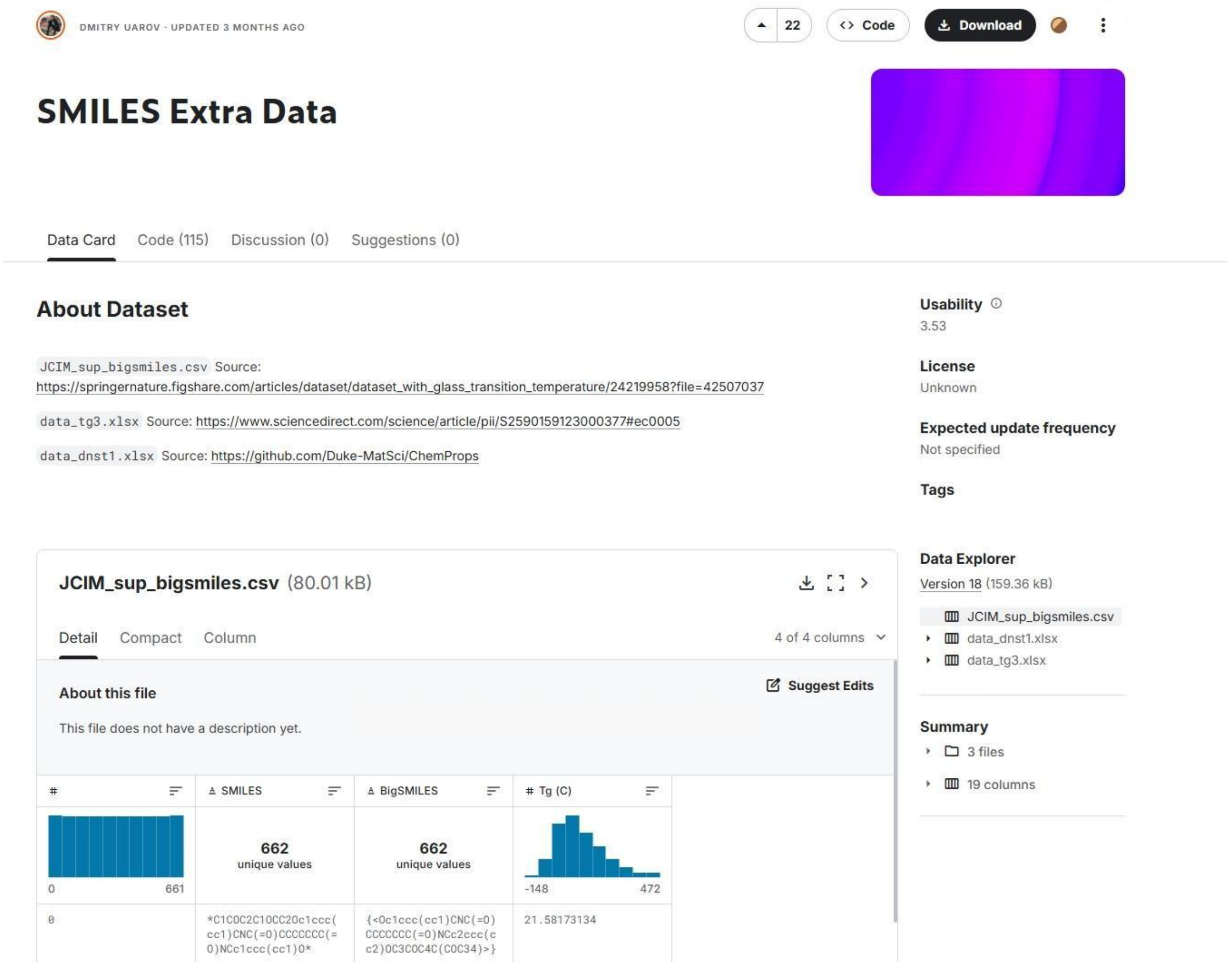
算法流程

Canonical SMILES 与外部数据



同一个分子SMILES
有很多种写法





算法流程

https://www.leskoff.com/s01812-0
https://www.kaggle.com/datasets/dmitryuarov/smiles-extra-data

分子图 & GNN

- 图构建：节点（原子序数/度/形式电荷/杂化/芳香/氢数/在环/质量），边（键型/在环/共轭/芳香）。
- 分子级特征：MolWt、HBD、HBA、TPSA、可旋转键数、SMILES 长度。
- 模型：GCN 堆叠 → GAT 注意力 → mean/max 池化 + 分子级特征 → 目标独立回归头（单任务 5 折）。

算法流程

```python
def smiles_to_graph(smiles):
    """SMILES文字列を分子グラフに変換"""
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None

        # 原子特徴量の取得
        atom_features = []
        for atom in mol.GetAtoms():
            features = [
                atom.GetAtomicNum(),
                atom.GetDegree(),
                atom.GetFormalCharge(),
                int(atom.GetHybridization()),
                int(atom.GetIsAromatic()),
                atom.GetTotalNumHs(),
                int(atom.IsInRing()),
                atom.GetMass()
            ]
            atom_features.append(features)

        if len(atom_features) == 0:
            return None

        # エッジ（結合）情報の取得
        edge_indices = []
        edge_features = []

        for bond in mol.GetBonds():
            i = bond.GetBeginAtomIdx()
            j = bond.GetEndAtomIdx()

            bond_features = [
                bond.GetBondTypeAsDouble(),
                int(bond.IsInRing()),
                int(bond.GetIsConjugated()),
                int(bond.GetIsAromatic())
            ]

            # 無向グラフとして扱う
            edge_indices.extend([[i, j], [j, i]])
            edge_features.extend([bond_features, bond_features])
```

```python
class PolymerGNN(nn.Module):
    def __init__(self, atom_features_dim=8, edge_features_dim=4, mol_features_dim=6,
                 hidden_dim=128, num_layers=2, num_targets=5, dropout=0.0):
        super(PolymerGNN, self).__init__()

        self.num_targets = num_targets
        self.dropout = dropout

        # Graph convolution layers
        self.convs = nn.ModuleList()
        self.batch_norms = nn.ModuleList()

        # 最初の層
        self.convs.append(GCNConv(atom_features_dim, hidden_dim))
        self.batch_norms.append(nn.BatchNorm1d(hidden_dim))

        # 中間層
        for _ in range(num_layers - 1):
            self.convs.append(GCNConv(hidden_dim, hidden_dim))
            self.batch_norms.append(nn.BatchNorm1d(hidden_dim))

        # 注意機構を追加
        self.attention = GATConv(hidden_dim, hidden_dim//4, heads=4, dropout=dropout)
        self.attention_bn = nn.BatchNorm1d(hidden_dim)

        # 分子レベル特徴量の処理
        self.mol_fc = nn.Sequential(
            nn.Linear(mol_features_dim, hidden_dim//2),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim//2, hidden_dim//2)
        )

        # 最終予測層（各ターゲット別）
        combined_dim = hidden_dim * 2 + hidden_dim//2  # mean + max pooling + mol features

        self.predictors = nn.ModuleList()
        for _ in range(num_targets):
            predictor = nn.Sequential(
                nn.Linear(combined_dim, hidden_dim),
                nn.ReLU(),
                nn.Dropout(dropout),
                nn.Linear(hidden_dim, hidden_dim//2),
                nn.ReLU(),
                nn.Dropout(dropout),
```

- NN_Density_fold_1_best.pt
- NN_Density_fold_2_best.p
- NN_Density_fold_3_best.p
- NN_Density_fold_4_best.p
- NN_Density_fold_5_best.p
- NN_FFV_fold_1_best.pth
- NN_FFV_fold_2_best.pth
- NN_FFV_fold_3_best.pth
- NN_FFV_fold_4_best.pth
- NN_FFV_fold_5_best.pth
- NN_Rg_fold_1_best.pth
- NN_Rg_fold_2_best.pth
- NN_Rg_fold_3_best.pth
- NN_Rg_fold_4_best.pth
- NN_Rg_fold_5_best.pth
- NN_Tc_fold_1_best.pth
- NN_Tc_fold_2_best.pth
- NN_Tc_fold_3_best.pth
- NN_Tc_fold_4_best.pth
- NN_Tc_fold_5_best.pth
- NN_Tg_fold_1_best.pth
- NN_Tg_fold_2_best.pth
- NN_Tg_fold_3_best.pth
- NN_Tg_fold_4_best.pth
- NN_Tg_fold_5_best.pth

算法流程

Mordred 描述符 & CatBoost

安装的话就pip install mordred

对于单个分子计算所有的描述符。注意下mordred对于有些无法计算的描述符会在dataframe里面返回欠损值并且写明原因，要丢到机器学习模型里面进行进一步计算之前一定要对这些NaN进行处理比如变0之类的。

```
from mordred import Calculator, descriptors
calc = Calculator(descriptors, ignore_3D=True)
X_mord = pd.DataFrame(calc.pandas([mol]))
X_mord
```

	ABC	ABCGG	nAcid	nBase	SpAbs_A	SpMax_A	SpDiam_A	SpAD_A	SpMAD_A	LogEE_A	...	SRW10	TSRW10	MW	AMW
0	5.656854	5.42766	0	0	10.424292	2.135779	4.271558	10.424292	1.303037	2.969338	...	8.298291	35.247635	106.041865	7.574419

模型线上推理时训练即可

```
def model(train_d,test_d,model,target,submission=False):
    # We divide the data into training and validation sets for m
    train_cols = set(train_d.columns) - {target}
    test_cols = set(test_d.columns)
   # Intersect the feature columns
    common_cols = list(train_cols & test_cols)
    X=train_d[common_cols].copy()
    y=train_d[target].copy()
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size

    Model=model(verbose=0)
    if submission==False:
       Model.fit(X_train,y_train,verbose=0)
       y_pred=Model.predict(X_test)
       return mean_absolute_error(y_pred,y_test)          # We as
    if submission==True:
       Model.fit(X,y)
       submission=Model.predict(test_d[common_cols].copy())
       return submission
```

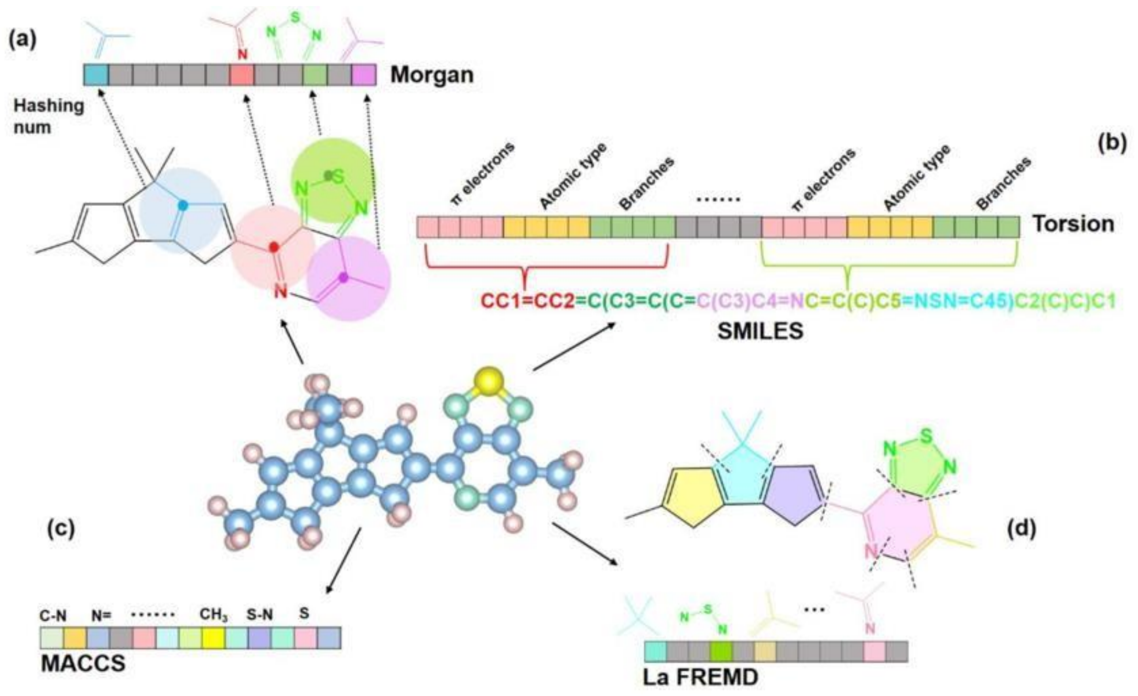
算法流程

https://zhuanlan.zhihu.com/p/349670859

指纹 + 图统计 & XGBoost

- 特征：Morgan(r=2,128bit) + MACCS(166) + RDKit 物化 + NetworkX 图统计（直径/最短路/环数)。
- 先验与筛选：按目标的特征白名单 + 方差阈值(0.01)裁剪低信息列。
- 增强：随机 SMILES 与 GMM 合成样本；各目标独立建模与调参。

算法流程




```python
def smiles_to_combined_fingerprints_with_descriptors(smiles_list, radius=2, n_bits=128):
    generator = GetMorganGenerator(radius=radius, fpSize=n_bits)
    atom_pair_gen = GetAtomPairGenerator(fpSize=n_bits)
    torsion_gen = GetTopologicalTorsionGenerator(fpSize=n_bits)

    fingerprints = []
    descriptors = []
    valid_smiles = []
    invalid_indices = []

    for i, smiles in enumerate(smiles_list):
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            # Fingerprints
            morgan_fp = generator.GetFingerprint(mol)
            #atom_pair_fp = atom_pair_gen.GetFingerprint(mol)
            #torsion_fp = torsion_gen.GetFingerprint(mol)
            maccs_fp = MACCSkeys.GenMACCSKeys(mol)

            combined_fp = np.concatenate([
                np.array(morgan_fp),
                #np.array(atom_pair_fp),
                #np.array(torsion_fp),
                np.array(maccs_fp)
            ])
            fingerprints.append(combined_fp)

            # RDKit Descriptors
            descriptor_values = {}
            for name, func in Descriptors.descList:
                try:
```

算法流程

加权融合

```
sub1 = pd.read_csv("gnn_sub.csv").sort_values(['id']).reset_index(drop=True)
sub2 = pd.read_csv("cat_sub.csv").sort_values(['id']).reset_index(drop=True)
sub3 = pd.read_csv("xgb_sub.csv").sort_values(['id']).reset_index(drop=True)
```

[12]

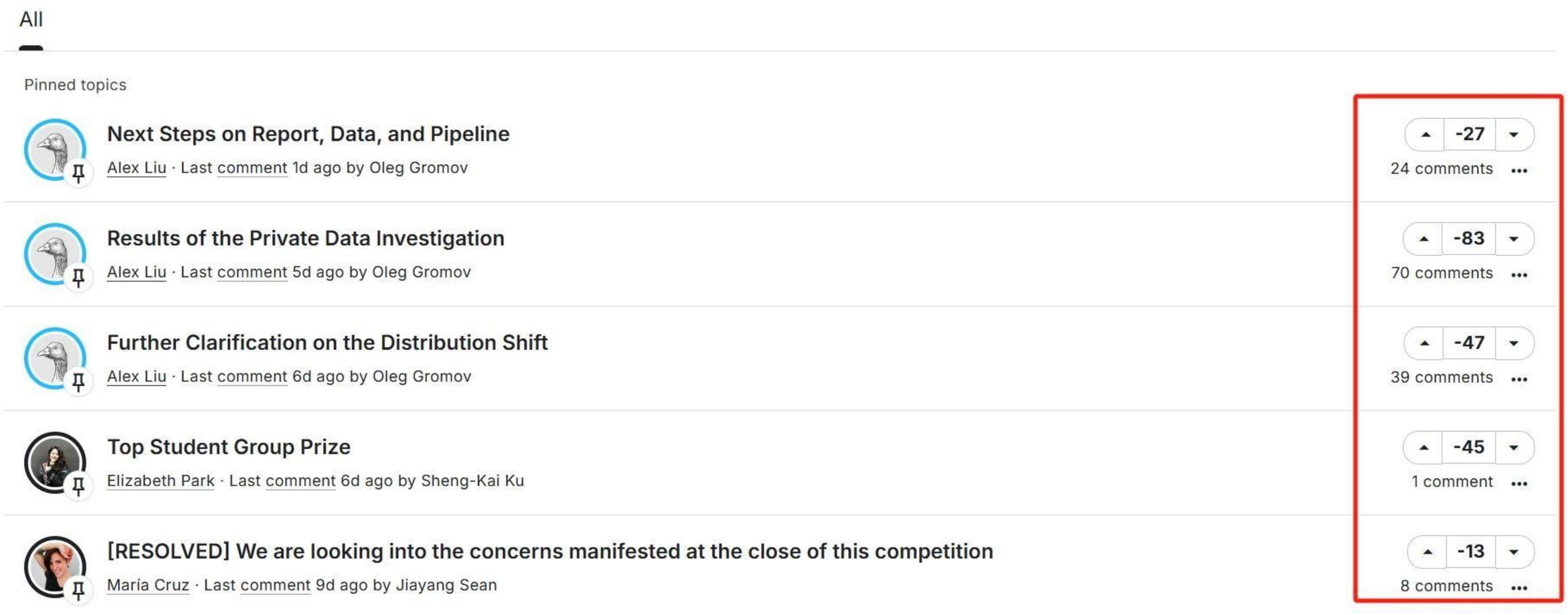
```
w_gnn = 0.4
w_cat = 0.3
w_xgb = 0.3
targets = ['Tg', 'FFV', 'Tc', 'Density', 'Rg']

# 融合
submission_fused = sub1.copy()
for col in targets:
    submission_fused[col] = w_gnn * sub1[col] + w_cat * sub2[col]+ w_xgb * sub3[col]

# 输出融合后的结果
submission_fused.to_csv('submission.csv', index=False)
submission_fused.head()
```

算法流程

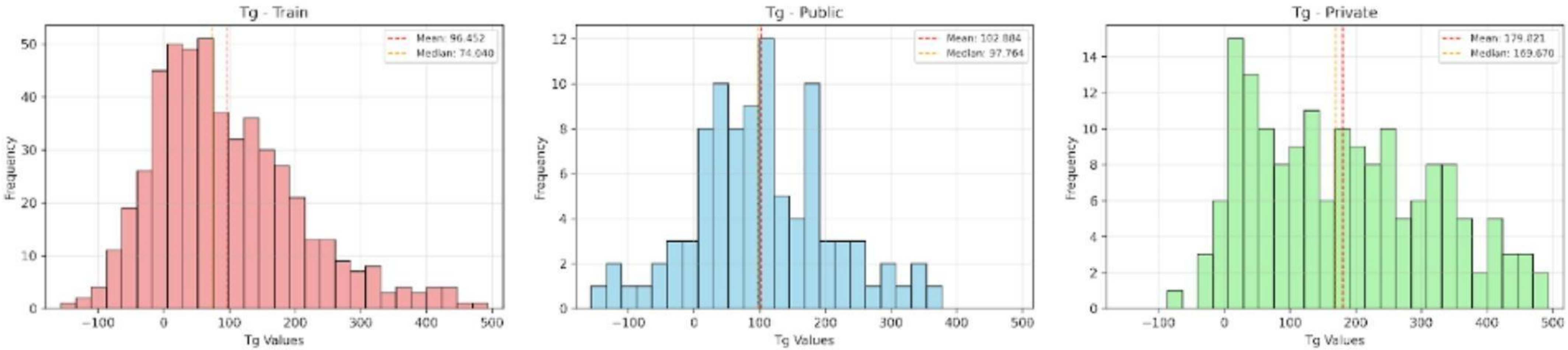
数据分布与 shakeup



算法流程

https://www.kaggle.com/competitions/neurips-open-polymer-prediction-2025/discussion/607784

数据分布与 shakeup



算法流程

NeurIPS: XGBoost-beasline - Tg by 9/5, then add 32 Succeeded (after deadline) · 10m ago · Notebook NeurIPS: XGBoost-beasline | Version 11	0.074	0.070
NeurIPS: XGBoost-beasline - Version 5 Succeeded · 7d ago · Notebook NeurIPS: XGBoost-beasline | Version 5	0.096	0.070
NeurIPS: XGBoost-beasline - Version 8 Succeeded · 6d ago · Notebook NeurIPS: XGBoost-beasline | Version 8	0.094	0.070

https://www.kaggle.com/competitions/neurips-open-polymer-prediction-2025/discussion/608289