**一、TabNet**

TabNet是由谷歌发布的模型：  
论文地址：<https://arxiv.org/abs/1908.07442>

PyTorch-TabNet是基于Pytorch实现的TabNet。  
Github：<https://github.com/dreamquark-ai/tabnet>  
官方文档：<https://dreamquark-ai.github.io/tabnet/>  
安装：pip install pytorch-tabnet  
导包：import pytorch\_tabnet

参考链接：[17.TabNet (PyTorch-TabNet) | Kaka Wan Yifan](https://kakawanyifan.com/10417)

数据集格式为.xlsx的表格类数据

二、SMOTE算法

SMOTE，全称为Synthetic Minority Over-sampling Technique，即合成少数类过采样技术，是一种用于处理不平衡数据集的算法。它通过在现有的少数类样本之间创建合成样本来增加少数类的数量，以此来平衡数据集。SMOTE的基本思想如下：

1. **选择样本**：随机选择一个少数类的样本。
2. **寻找邻居**：找出选定样本的最近邻居（通常是k个邻居，k的选择取决于用户）。
3. **合成样本**：从这些邻居中随机选择一个，然后在选定样本与邻居样本之间的连线上随机选择一个点，生成一个新的合成样本。

SMOTE的变体主要包括：

1. **Borderline-SMOTE**：这种变体只关注边界样本，即那些位于类边界上的样本，而不是所有的少数类样本。这有助于减少在远离类中心的区域生成的噪声样本。
2. **Safe-Level-SMOTE**：这种变体尝试在生成合成样本时保持决策边界的稳定性，通过限制合成样本的生成数量来避免过拟合。
3. **ADASYN**：ADASYN（Adaptive Synthetic Sampling Approach）是SMOTE的一种改进，它根据每个少数类样本的局部属性来调整过采样的程度，而不是对所有样本使用固定的k值。
4. **KMeans-SMOTE**：这种变体首先使用k-means聚类算法对少数类样本进行聚类，然后在每个聚类中应用SMOTE，以更好地捕捉数据的局部特性。
5. **SVM-SMOTE**：结合了支持向量机（SVM）和SMOTE，首先使用SVM找到边界样本，然后仅在这些边界样本上应用SMOTE。
6. **Feature-Weighted SMOTE**：这种变体考虑了不同特征的重要性，可能会在生成合成样本时给予某些特征更高的权重。

模型中使用了BorderlineSMOTE。

**三、FocalLoss**

Focal Loss 是一种专门为解决类别不平衡问题设计的损失函数，由 Microsoft Research Asia 的 Lin et al. 在 2017 年的论文《Focal Loss for Dense Object Detection》中提出。它主要用于目标检测任务，但也适用于其他需要处理类别不平衡的场景。

Focal Loss通过减少对简单样本（即模型已经正确预测的样本）的关注，并增加对困难样本（即模型预测错误或不确定的样本）的关注。这是通过引入一个调节因子（modulating factor）来实现的，该因子基于模型对当前样本的预测准确性。

Focal Loss 的公式定义如下：



其中：

* *L* 是损失函数。
* *pt*​ 是模型对于实际类别的预测概率，对于正样本，*pt*​=*p*(*y*=1)；对于负样本，*pt*​=*p*(*y*=0)。
* *αt*​ 是平衡正负样本的权重系数，对于正样本是 *α*，对于负样本是 ((1−*α*)。
* *γ* 是调节因子，用于减少对简单样本的关注，通常设置为 2 或 5。

**特点**

* **关注困难样本**：通过  (1−*pt*​)*γ* 项，当 *pt*​ 接近 1 时，损失会减小，这样模型就会更加关注那些 *pt*​ 较小的困难样本。
* **类别不平衡处理**：通过 *αt*​ 项，可以为不同类别的样本分配不同的权重，从而更好地处理类别不平衡问题。
* **灵活性**：*α* 和 *γ* 参数可以根据具体任务进行调整，以获得最佳性能。

**四、PSO优化算法**

PSO（Particle Swarm Optimization，粒子群优化）算法是一种基于群体智能的优化算法，由Eberhart和Kennedy在1995年提出。它受到鸟群觅食行为的启发，通过模拟鸟群或鱼群等生物群体的社会行为来进行问题求解。

**基本原理**

PSO算法中，每个解被视为搜索空间中的一个“粒子”，每个粒子代表了问题的潜在解。粒子在搜索空间中飞行，通过跟踪两个“极值”来更新自己的位置和速度：

* **个体极值**：粒子自身所找到的最优解。
* **全局极值**：整个粒子群中所有粒子所找到的最优解。

**算法步骤**

1. **初始化**：随机初始化一群粒子的位置和速度。
2. **评价**：计算每个粒子的适应度值，即问题的目标函数值。
3. **更新个体极值**：如果当前粒子的适应度比个体极值好，就用当前位置更新个体极值。
4. **更新全局极值**：如果当前粒子的适应度比全局极值好，就用当前位置更新全局极值。
5. **更新速度和位置**：根据个体极值和全局极值更新每个粒子的速度和位置。
6. **迭代**：重复步骤2-5，直到满足停止条件（例如达到最大迭代次数或解的质量达到预设阈值）