模型使用说明

首先，在系统中安装 plink 和 Beagle 软件，对原始 vcf 文件（以 filter_269_indel.vcf 为例）进行清洗：

#1.vcf 转 plink 文件

plink --vcf filter_269_indel.vcf --double-id --allow-extra-chr --recode --out 269_indel1

#2.检查基因型和样本缺失情况

plink --file 269_indel1 --allow-extra-chr --missing

#3.先对基因型进行过滤

plink --file 269_indel1 --allow-extra-chr --geno 0.1 --recode --out 269_indel2

#4.接着对个体进行缺失控制

plink --file 269_indel2 --allow-extra-chr --mind 0.2 --recode --out 269_indel3

#5.次等位基因频率（MAF）控制，首先对基因进行检验

plink --file 269_indel3 --allow-extra-chr --freq

#6.过滤基因频率小于 0.05 的位点

plink --file 269_indel3 --allow-extra-chr --maf 0.05 --recode --out 269_indel4

#7.将 plink 文件转化为 vcf 后进行基因型填充

plink --file 269_indel4 --allow-extra-chr --recode vcf-iid --out 269_indel_GENO_I

#8.填充

Java -Xmx512g -jar beagle.22Jul22.46e.jar gt=269_indel_GENO_I.vcf out=imputed_269_indel nthreads=64

beagle.22Jul22.46e.jar 见：LGB\LGB-main\beagle.22Jul22.46e.jar

#8.解压 imputed_269_indel.vcf.gz

gunzip imputed_269_indel.vcf.gz

#9.对 vcf 文件转化为 012 文件(raw 格式)

plink --vcf imputed_269_indel.vcf --double-id --recodeA --allow-extra-chr --out 269_indel_geno0.1_maf0.05_mind0.2_imputed_01

清洗完成后，从得到的 raw 文件中筛选出 LGB\LGB-mian 中对应的位点，使用对应的 module 文件进行预测即可。（以下是范例展示）

import lightgbm as lgb

import numpy as np

```
from sklearn.datasets import make_regression
from sklearn.metrics import mean_squared_error
from scipy.stats import pearsonr
import pandas as pd
import joblib
# 1. 准备预测数据和真实标签
# 生成示例回归数据用于预测
X_pred=pd.read_csv("/vochome/ZTY/jupyter/work/tao_ML/indel/indel_train_data/ZJ_apart_from_tao_indel.csv")
Y_True = pd.read_csv("/vochome/ZTY/jupyter/work/tao_ML/pheno/ZJ_pheno.csv")
X_pred=X_pred.iloc[:,2:]
list = ['citric_acid','malic_acid']

for i in list:
    y_True = Y_True[i]
    try:
        # 加载本地保存的模型文件

        model_path=f"/vochome/ZTY/jupyter/work/tao_ML/LGB/model/apart_from_tao_indel_model/LGB_indel_{i}.joblib"
        model = joblib.load(model_path)
    except FileNotFoundError:
        print("模型文件未找到，请检查文件路径。")
        exit(1)
    y_pred = model.predict(X_pred)

    # 4. 计算均方误差
    mse = mean_squared_error(y_True, y_pred)

    # 5. 计算皮尔逊相关系数
    pcc, p_value = pearsonr(y_True, y_pred)
```

```
# 6. 计算相对误差
mre = np.mean(np.abs((y_pred-y_True)/y_True))

# 7. 输出结果
print(f"{i}的预测的回归值:", y_pred)
print(f"{i}的均方误差 (MSE):", mse)
print(f"{i}的皮尔逊相关系数:", pcc)
print(f"{i}的皮尔逊相关系数 p 值:", p_value)
print(f"{i}的平均相对误差：",mre)
```