代码说明与整体思路介绍

数据准备

1. 赛方提供数据
    - train.csv: 训练集交易数据
    - test.csv: 测试集交易数据
    - 位置: data/train.csv, data/test.csv

3. 额外爬取的补充数据
    - 股票基本信息：上市时间、行业分类等（该数据通过akshare网站进行爬取）
    - 位置: code/data/additional/all_stock_info_df.joblib
    - 财报信息，包括营业额、利润率、财报发布时间等（该数据通过akshare网站进行爬取）

数据处理与特征工程流程 (stage0+stage1+stage2+stage3)

0. 基于赛方数据推导计算金融因子
    - Alpha158: Qlib库中的技术分析因子
    - Alpha360: Qlib库中的另一组技术指标
    - 位置: temp/qlib_alpha158_ranked.pkl, temp/qlib_alpha360_ranked.pkl
1. 股票信息合并（stage1）
    - 将赛方给的数据与我们收集的股票基本信息、财报信息合并
2. 加入相关信息（stage2）
    - 构造lag特征
    - 加入了日期相关信息（如星期几；是否节假日等）
3. 数据处理
    - 将特征因子进行转换分布，使之更易于被模型拟合
    - 将股票的特定特征，包括涨跌、涨跌幅度、是否是前10或者后10等特征进行lag操作（滞后操作）
4. 特征选择

- 使用3种检验方法投票综合决定6个任务应该使用哪些特征。

5. 数据分割
    - 将处理好的大表按照时间拆分成训练、验证、测试数据集
    - 针对不同模型进行了不同的数据处理。

模型训练流程 (stage4)

1. 学习任务

```
- 涨跌排名靠前靠后二分类任务
```

- 是否在前十后十不平衡二分类任务
- 涨跌幅相对排名值回归预测任务
- 收盘价回归预测任务

2. 主要使用模型
    - 表格机器学习
    - 深度学习
        -
    - TemporalFusionTransformer等模型
    - 统计学习
    - ARIMA、ETS等模型
3. 集成学习
    - RandomForest、XGBoost等本身为集成模型
    - 每一个任务本身是Autogluon使用了Weighting、Stacking、Bagging等技巧
    - 不同任务之间进行了stage6

环境配置

注意，由于AutoGluon的Docker镜像过大，虽然如下的DockerFile可以完美构建我们的镜像，但是我们决定在init.sh中安装相关依赖！

```
# 1. 选择合适的基础镜像
FROM autogluon/autogluon:1.3.1-cuda12.4-jupyter-ubuntu22.04-py3.11

# 2. 设置工作目录
WORKDIR /app

# 3. 复制文件到容器
COPY requirements.txt /app/
COPY third_parties/ /app/third_parties/

# 4. 安装 Python 依赖
# ENV PIP_INDEX_URL=https://pypi.tuna.tsinghua.edu.cn/simple
# RUN pip install --upgrade pip -i https://pypi.tuna.tsinghua.edu.cn/simple
RUN pip install --no-cache-dir -r requirements.txt -i
https://pypi.tuna.tsinghua.edu.cn/simple



# thundersvm
# cmake -DCMAKE_C_COMPILER=gcc-11 -DCMAKE_CXX_COMPILER=g++-11 ..
```

```
# cd ../python
# python setup.py install
# sudo apt-get install libnuma-dev

# 安装 flash-attn


COPY ./code/ /app/code/
# COPY ./model/ /app/model/
COPY ./init.sh /app/init.sh
COPY ./train.sh /app/train.sh
COPY ./test.sh /app/test.sh
COPY ./readme.pdf /app/readme.pdf

EXPOSE 8888
ENV JUPYTER_WORKSPACE_NAME=app
CMD ["jupyter", "lab", "--ip=0.0.0.0", "--allow-root", "--no-browser", "--port=8888"]
```

由于比赛官方给的nvidia镜像太大，下载太慢，我们未能构建成功。

实际上我们的python代码非常鲁棒，在3.10\3.11\3.12的conda环境下，都能正确在8GB显存下跑通。

我们决定使用python 3.10 slim ，在init.sh中动态安装依赖。

数据

使用了 akshare 公开数据，数据获取链接为(https://github.com/akfamily/akshare)，在训练以上模型时均使用

预训练模型

1. 模型名称： TabPFN (Foundation Model for Tabular Data)

开源链接地址： https://github.com/PriorLabs/TabPFN

文件名及MD5值：

16bdb6b7041d4cbbe81f7d9153b00422 tabpfn-v2-classifier.ckpt

13bd1cee380f5ac2ec1da942bba7a910 tabpfen-v2-regressor.ckpt

2. 模型名称： TabPFNMix Classifier

开源链接地址： https://huggingface.co/autogluon/tabpfn-mix-1.0-classifier

说明： 此模型包含多个文件，我们对整个模型目录进行打包计算MD5值。

获取方式及MD5值：

运行命令 tar -cf - data/pretrained/autogluon/tabpfn-mix-1.0-classifier | md5sum

输出结果: 8d4b680dd7e98e0c859d8f868cef8d4b -

3. 模型名称： AutoGluon - Chronos 系列模型

开源链接地址： https://huggingface.co/collections/amazon/chronos-models-and-datasets-65f1791d630a8d57cb718444

模型文件及MD5值说明： 由于我们使用了 Chronos 系列的多个预训练模型，为每一个模型文件单独提供MD5值不切实际。为保证复现的准确性，我们提供以下可复现的批量下载方式。

依赖库版本： huggingface-hub==0.23.4

复现下载脚本：

```
from huggingface_hub import snapshot_download

model_ids = [
    "amazon/chronos-t5-tiny",
    "amazon/chronos-t5-small",
    "amazon/chronos-t5-base",
]

for model_id in model_ids:
    print(f"Downloading {model_id}...")
    snapshot_download(
        repo_id=model_id,
        local_dir=f"./models/{model_id.replace('/', '_')}",
        local_dir_use_symlinks=False
    )
print("All Chronos models downloaded.")
```

说明： 在指定 huggingface-hub 版本下运行以上Python代码，即可下载我们使用的全部Chronos模型文件。如图所示，从开源链接网页中可以看到，这一系列的模型在2025年5月1日到邮件发送日期期间，并无更新，因而下载的版本是早已发布开源的（2025年2月17日）。

算法以及方法的创新点

本研究并未依赖单一模型或单一路径，而是构建了一个多任务、多模型、多验证阶段的系统化框架，以数据驱动的方式确定最优建模策略。