大作业报告

研究背景

在金融市场中，波动性是一个至关重要的概念，它描述了资产价格在一定时间内的变动幅度，直接关系到投资者的风险评估和投资决策。高波动性通常与市场动荡和价格大幅波动的时期相关联，而低波动性则描述了更为平静和稳定的市场。在金融市场中，波动性高意味着价格变动剧烈，这可能是由于市场信息的快速变化、宏观经济因素的波动或是市场参与者情绪的不稳定。这种高波动性可能会为投资者带来更高的风险，但同时也可能带来更高的回报机会。相反，低波动性通常意味着市场更加稳定，价格变动较小，这为投资者提供了一个相对可预测的投资环境。然而，低波动性也可能意味着市场活力不足，潜在的回报较低。

对于交易公司，如Optiver，波动性的预测尤为关键。作为全球领先的电子市场制造商，Optiver专注于提供流动性和改善市场效率，其业务涵盖了期权、ETFs、现金股票、债券和外汇等多个领域。波动性的准确预测能够帮助公司制定更合理的交易策略，为投资者提供更公平的交易价格，从而在竞争激烈的市场中保持领先地位。此外，波动性也是衍生品定价的关键因素之一，尤其是期权定价。期权的价值在很大程度上取决于标的资产的预期波动性。因此，对波动性的深入理解和准确预测对于期权交易者来说至关重要，它直接影响到期权的买卖决策和风险管理。

因此，本文以kaggle平台上的一场由Optiver公司发起的实际波动性预测问题为背景，利用该竞赛提供的公开数据集，采用3种机器学习模型对波动性进行预测和分析，完整项目代码位于github repo。

数据集介绍

数据集由Optiver公司提供，包含与金融市场交易相关的股票市场数据，通过一秒钟的分辨率，提供了对现代金融市场微观结构的独特细粒度观察。数据集的总大小为2.73 GB，包含229个文件，类型为Parquet和CSV，可由此下载，其具体包含如下内容：

1. 订单数据 book_[train/test].parquet：按`stock_id`分区的Parquet文件，提供了市场上最具竞争力的买卖订单数据。数据集中包含了订单的前两级，第一级的价格更具竞争力，并且会优先执行。每条记录包括：
    - stock_id: 股票的ID代码。
    - time_id: 时间桶的ID代码。
    - seconds_in_bucket: 从时间桶开始时刻至当前时刻的秒数。
    - bid_price[1/2]: 最/第二有竞争力的买方价格。
    - ask_price[1/2]: 最/第二有竞争力的卖方价格。
    - bid_size[1/2]: 最/第二有竞争力的买方订单的股票数量。
    - ask_size[1/2]: 最/第二有竞争力的卖方订单的股票数量。
2. 交易数据 trade_[train/test].parquet: 按`stock_id`分区的Parquet文件，包含实际执行的交易数据。通常市场上的被动买卖意向更新（订单更新）比实际交易要多，因此该文件比订单文件更稀疏。每条记录包括：
    - stock_id: 股票的ID代码。
    - time_id: 时间桶的ID代码。
    - seconds_in_bucket: 从时间桶开始时刻至当前时刻的秒数。注意，由于交易和订单数据来自相同的时间窗口，且交易数据通常更稀疏，此字段不一定从0开始。
    - price: 一秒内执行交易的平均价格，价格已标准化，并且按每笔交易的股份数量加权平均。
    - size: 交易的股票总数。

- order_count: 发生的独立交易订单数量。

3. 训练集真实值 train.csv：提供训练集的真实值，包括：
   - stock_id: 股票的ID代码。
   - time_id: 时间桶的ID代码。
   - target: 预测的目标，在与特征数据相同的`stock_id`/`time_id`下，接下来的10分钟窗口内计算出的实际波动率。特征数据和目标数据之间没有重叠。
4. 测试集数据 test.csv: 公开的测试数据，包括：
   - stock_id: 股票的ID代码。
   - time_id: 时间桶的ID代码。
   - row_id: 提交行的唯一标识符。

注：这是一个代码竞赛，只有测试集的前三行可供下载，更多的测试数据进行了隐藏。并且由于比赛已经结束，无法进行代码结果的评估。因此，本次实验中，并没有采用提供的测试集，而是将提供的训练数据集按比例随机分为训练集、验证集和测试集进行模型的训练、验证和测试。

评价指标

按照竞赛给出的要求，结果使用相对均方根误差（Root Mean Square Percentage Error）进行评价，定义如下：

$$\mathrm{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}((y_i - \hat{y}_i)/y_i)^2}$$

特征工程

由于金融数据的信噪比较高，并且各项数据间的量纲不同，因此首先对数据进行预处理，利用原始数据计算相关的金融指标，并对指标的均值、方差、极值、和等进行计算，并进行PCA降维，将降维后的特征作为模型的输入特征，该部分对应代码文件feature_engineering.py，金融相关指标参考竞赛官方提供的tutorial。

加权平均价格

订单是股票估值的主要来源之一。合理的基于订单的估值必须考虑两个因素：订单的水平和规模，可以使用加权平均价格（WAP, weighted averaged price）来计算瞬时股票估值，定义为：

$$\mathrm{WAP}_i = \frac{\mathrm{BidPrice}_i \times \mathrm{AskSize}_i + \mathrm{AskPrice}_i \times \mathrm{BidSize}_i}{\mathrm{BidSize}_i + \mathrm{AskSize}_i}$$

其中，$i = 1\ or\ 2$代表第一或第二竞争力的订单。如果两份订单的买卖报价分别处于相同的价格水平，则报价较多的订单将产生较低的股票估值，因为该订单会有更多的意向卖家，而更多的卖家意味着市场上供应更多的事实，从而导致股票估值较低。另外，本文使用相同股票的最具竞争力和第二竞争力的WAP差来描述两种竞争力订单引入的估值差异。

对数回报率

对数回报率用来比较股票两个时刻间的差异，定义为：

$$r_{t_1,t_2} = \log(\frac{S_{t_2}}{S_{t_1}})$$

其中，S_t是股票在t时刻的价格，利用加权平均价格进行计算。

实际波动率

实际波动率通过对数回报率进行计算，定义为：

$$\sigma = \sqrt{\sum_t r_{t-1,t}^2}$$

除了计算整个时间桶内的实际波动率，还对时间桶进行基于窗口实际波动率计算，分别对时间桶内时间大于$S \in 100, 200, 300, 400, 500$ 的部分进行实际波动率计算，作为输入特征。

其它

除了上述几个重要的特征，本文还使用了价格差、买卖差、股票交易总股数等常见金融指标，并计算每项指标的均值、方差、和、最大值、最小值。为了减少输入特征的冗余，本文还使用了主成分分析（PCA）对输入特征进行降维。

机器学习模型

本文选择了两种树模型和一种感知机模型来进行实际波动率预测。

XGBoost

XGBoost（Extreme Gradient Boosting）是一种高效且灵活的梯度提升框架，通过迭代地将多个弱学习器（通常是决策树）结合在一起，以减少预测误差，从而提高模型的准确性和泛化能力。它采用L1和L2正则化来防止过拟合，并行计算提高训练速度，能够高效处理稀疏数据，并支持自定义损失函数。其核心思想是在每一步迭代中，通过拟合残差来更新模型，使得最终预测结果更加精确。

模型的目标函数为：

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k)$$

第一项描述了拟合残差，第二项为正则化项，定义为：

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2$$

其中，γ和λ均为正则化参数，T是树的叶子节点数，ω_j是叶子节点的权重。

每一轮的预测结果由加法模型定义：

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

其中，f_t是第t轮的弱学习器。

LightGBM

LightGBM（Light Gradient Boosting Machine）是一个高效的梯度提升框架，与XGBoost相比，它通过基于**直方图算法**来加速训练过程，并采用**基于叶子节点的最优拆分方式**构建决策树，从而提高了训练速度和内存效率，更适合大规模数据集和高维特征的处理。

直方图算法

LightGBM采用了基于直方图的决策树构建算法，这种方法通过将连续特征值离散化为有限数量的离散值（即直方图的bins），从而显著减少了计算量。具体步骤如下：

1. **离散化特征**：首先，将连续特征值根据预设的bin数进行离散化。例如，如果设定的bin数为256，那么每个特征会被划分成256个区间，每个区间包含特征值的一部分。
2. **构建直方图**：在每个节点的分裂过程中，为每个特征构建一个直方图，统计每个bin中样本的梯度和二阶梯度累加值。
3. **寻找最佳分裂点**：通过遍历直方图中的bin，找到使损失函数下降最多的分裂点。由于bin的数量远小于原始特征值的数量，这一步大大加速了最佳分裂点的搜索过程。

基于叶子节点的最优拆分

与传统的按层级生长树结构不同，LightGBM采用了基于叶子节点的最优拆分策略，即每次选择增益最大的叶子节点进行拆分，而不是对每一层的所有节点进行拆分。具体过程如下：

1. **初始化**：从根节点开始，计算所有特征的所有可能分裂点的增益，选择增益最大的分裂点进行分裂。
2. **逐步拆分**：每次迭代选择当前所有叶子节点中，增益最大的节点进行拆分，继续计算其子节点的分裂点和增益
3. **迭代停止**：重复上述步骤，直到满足预设的停止条件，如达到最大深度或叶子节点数。

MLP

多层感知机（MLP）是一种前馈神经网络，由一个输入层、一个或多个隐含层和一个输出层组成。每层包含若干神经元，每个神经元通过权重连接到下一层的神经元。MLP利用非线性激活函数（如ReLU、sigmoid、tanh）将输入数据通过各层进行变换，最终输出预测结果。其训练过程通常使用反向传播算法，通过最小化损失函数来调整权重。

1. 前向传播

$$
\begin{aligned}
z^{(l)} &= W^{(l)} a^{(l-1)} + b^{(l)} \\
a^{(l)} &= \sigma(z^{(l)})
\end{aligned}
$$

其中，$z^{(l)}$是第l层的线性变换结果，$W^{(l)}$是权重矩阵，$b^{(l)}$是偏置，$a^{(l)}$是激活后的输出，σ是激活函数。

2. 反向传播（损失函数L对权重的梯度）

$$
\begin{aligned}
\delta^{(l)} = \frac{\partial L}{\partial z^{(l)}} &= (\delta^{(l+1)} W^{(l+1)}) \odot \sigma'(z^{(l)}) \\
\frac{\partial L}{\partial W^{(l)}} &= \delta^{(l)} (a^{(l-1)})^T \\
\frac{\partial L}{\partial b^{(l)}} &= \delta^{(l)}
\end{aligned}
$$

其中，$\delta^{(l)}$是第l层的误差项，⊙表示逐元素相乘，σ'是激活函数的导数。

多层感知机（MLP）的优点在于其强大的非线性特征学习能力，使其能够处理复杂的模式识别和回归任务。由于其多层结构和灵活的激活函数，MLP可以近似任何连续函数，具有很高的表达能力。

实验与结果分析

XGBoost、LightGBM与MLP的定义分别对应xgb_model.py、lgb_model.py与mlp_model.py文件，对应的模型配置与超参设置分别在xgb_config.yaml、lgb_config.yaml与mlp_config.yaml文件中。

实验平台

Thinkbook 14+ 锐龙笔记本，CPU为AMD R7-6800H，无独显，实验均使用CPU进行。

输入特征

进行PCA降维前的特征数为 ，PCA的主成分贡献率设置为0.9，降维后的特征数量为。

利用joblib库进行数据的并行读取，之后利用单核和pandas包进行特征计算，特征工程包括数据读取与特征计算）总耗费时间为238.47s。

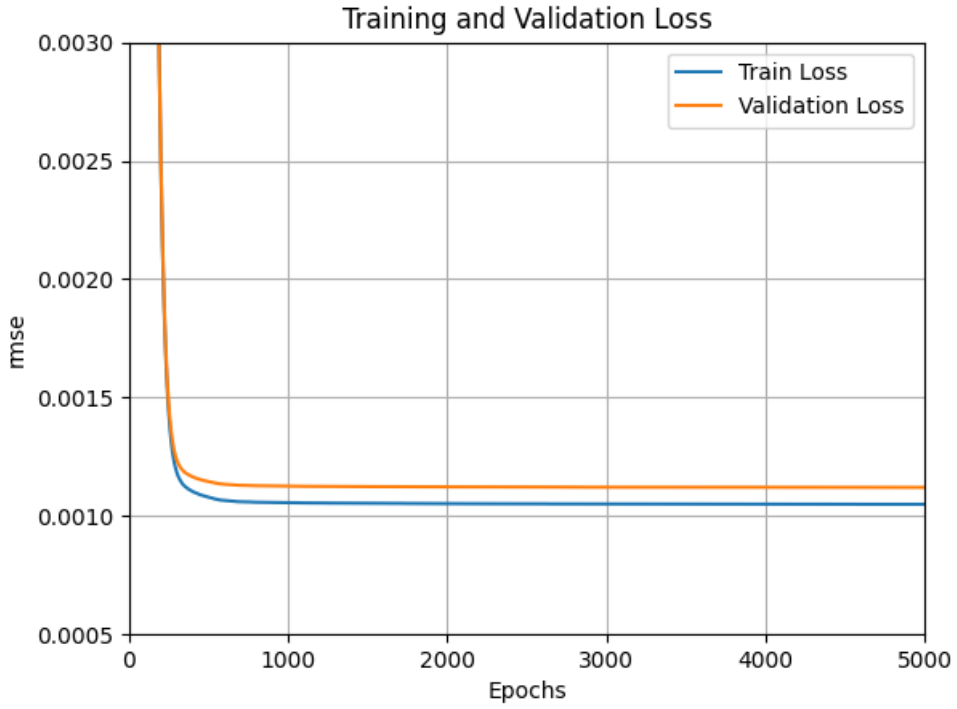
对于XGBoost和LightGBM，输入特征去除了`row_id`，保留其它输入特征，未进行归一化；对于MLP，输入特征去除了`row_id`，并将`time_id`和`stock_id`进行了embedding操作，并与其它特征进行concatenate，归一化后作为MLP的输入。

竞赛官网只有测试集的前三行可供下载，更多的测试数据进行了隐藏。并且由于比赛已经结束，无法进行代码结果的评估。因此，本次实验中，并没有采用提供的测试集，而是将提供的训练数据集按比例8：1：1随机分为训练集、验证集和测试集进行模型的训练、验证和测试。

实验结果

XGBoost

XGBoost模型的训练和验证loss曲线如下图：



训练总耗时为：372.82s（5000rounds），测试集的rmspe得分为：0.2467。

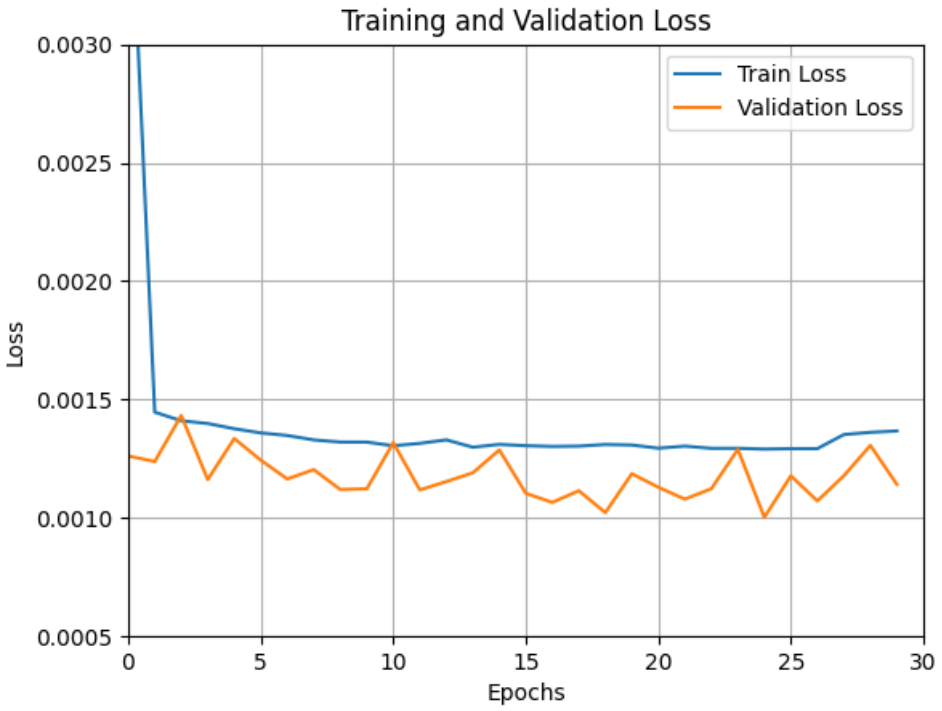
LightGBM

LightGBM模型的训练和验证loss曲线如下图：

训练总耗时为：200.08s（5000rounds），测试集的rmspe得分为：0.2321。

MLP

LightGBM模型的训练和验证loss曲线如下图：



训练总耗时为：687.72s（30epochs），测试集的rmspe得分为：0.2213。

结果分析

在对金融市场波动性预测的实验中，本文采用了XGBoost、LightGBM和MLP三种机器学习模型进行了比较分析。实验结果显示，尽管XGBoost和LightGBM都属于高效的梯度提升框架，但LightGBM在训练速度和测试集的预测精度上均优于XGBoost，这得益于其基于直方图的决策树构建算法和基于叶子节点的最优拆分策略，这些技术显著提高了处理速度。

进一步对比XGBoost和LightGBM发现，尽管XGBoost在迭代中通过拟合残差来更新模型，但LightGBM通过其独特的算法优化，在相同迭代轮数下达到了更低的RMSPE得分，显示出在大规模数据集上更优的性能。此外，LightGBM的训练时间也显著短于XGBoost，这在需要快速模型迭代和实时预测的应用场景中尤为重要。

将决策树模型与MLP进行对比，MLP虽然在训练时间上最长，但其在测试集上展现出了最低的RMSPE得分，这表明MLP在处理复杂非线性回归任务时的强大能力。MLP的多层结构和灵活的激活函数使其能够近似任何连续函数，具有很高的表达能力，这可能是其在预测精度上超越决策树模型的关键因素。

综合考虑，本次实验的结论是，在金融市场波动性预测任务中，虽然XGBoost和LightGBM在训练效率上表现出色，但MLP在预测精度上具有明显优势。选择最合适的模型需要根据实际应用场景的需求，包括预测精度、训练时间、资源消耗等因素进行权衡。对于需要快速响应的交易环境，LightGBM可能是更合适的选择；而对于追求最高预测精度的应用，MLP可能更值得考虑。