



北京大学量化交易协会2021级培训

Kaggle波动率预测比赛专题研究

叶文轩 刘君瑶 王冠 詹缘 张肖纬 吴昊雨 郑昕然 2021-10-25

Kaggle波动率预测比赛专题研究

1 Kaggle比赛简介

2 特征工程

3 模型介绍

4 PB 2nd方案详解



Kaggle 比赛简介

赛事简介

2021.6.28-9.27，Optiver在Kaggle举办的波动率预测比赛，所有参赛模型都将在训练平台和实盘数据上试验打分。打分标准采用 $RMSPE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\frac{y_i - \hat{y}_i}{y_i})^2}$ ，排名靠前的队伍可以获得不同程度的奖金。Kaggle平台提供了代码共享区，以及讨论区。同时平台还提供了GPU以供代码运行

Featured Code Competition

Optiver Realized Volatility Prediction

Apply your data science skills to make financial markets better

Optiver

3,852 teams

3 months to go

\$100,000
Prize Money

Overview

Data

Code

Discussion

Leaderboard

Rules

Overview

Description

Evaluation

Timeline

Prizes

Code Requirements

Volatility is one of the most prominent terms you'll hear on any trading floor – and for good reason. In financial markets, volatility captures the amount of fluctuation in prices. High volatility is associated to periods of market turbulence and to large price swings, while low volatility describes more calm and quiet markets. For trading firms like Optiver, accurately predicting volatility is essential for the trading of options, whose price is **directly related to the volatility** of the underlying product.

As a leading global electronic market maker, Optiver is dedicated to continuously improving financial markets, creating better access and prices for options, ETFs, cash equities, bonds and foreign currencies on numerous exchanges around the world. Optiver's teams have spent countless hours building

kaggle

Create

Home

Competitions

Datasets

Code

Discussions

Courses

More

Search

Competitions

Grow your data science skills by competing in our exciting competitions. Find help in the [documentation](#) or learn about [InClass competitions](#).

+ Host a Competition

Your Work

Search competitions

All competitions

Entered

Hosted

Featured

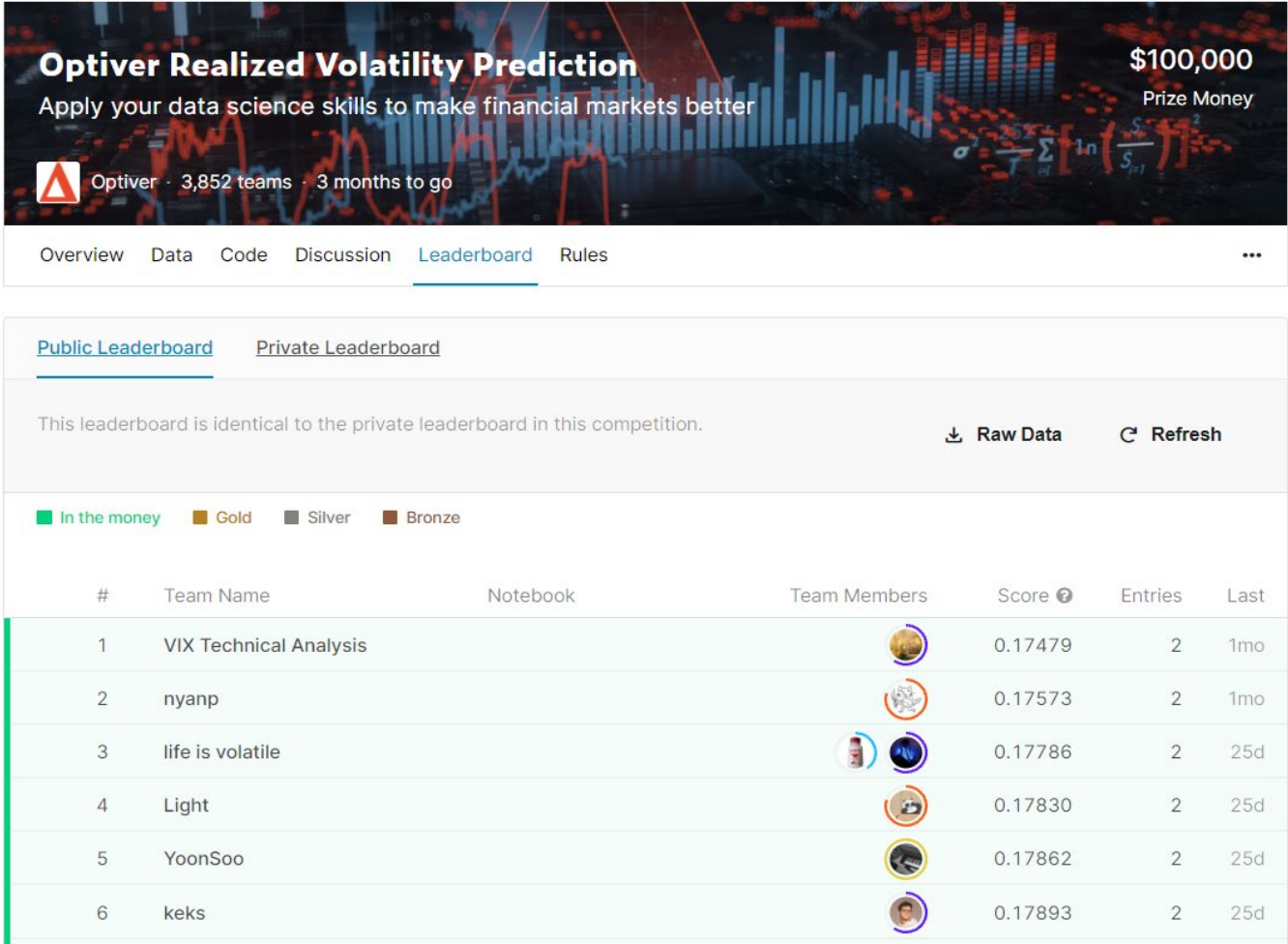
Research

Getting Started



Kaggle 比赛简介

获胜者决定方式



比赛有public leaderboard和private leaderboard 两种排行榜，他们分别基于public test set和private test set两个测试集，其中public leaderboard的排名一直在网站公布，private leaderboard会随着更多真实数据的加入而不断更新。

Potential winner完全由private leaderboard上的排名决定。如果出现平局，最先进入比赛的提交将成为获胜者。如果潜在获胜者因任何原因被取消资格，则获得下一个最高分排名的选手将被选为潜在获胜者。

数据简介

Order Book(订单簿)

- Order Book是指股票（或其他资产）交易时按照价格优先的顺序将买卖订单进行排序的订单序列，每个价格水平都有对应的买卖订单的数量。
- 买一和卖一总有一定的价差，此时交易无法进行。如果有买的人出价等于卖一的价格，就成交了，这种称为外盘，如果有卖的人出价等于买一的价格也能成交，称为内盘
- Ask: 表示卖量， Bid: 表示买量

bid #	price	ask #
	151	196
	150	189
	149	148
	148	221
251	147	
321	146	
300	145	
20	144	

以148的价格发一笔
交易量为20的买单



bid #	price	ask #
	151	196
	150	189
	149	148
	148	201
251	147	
321	146	
300	145	
20	144	

数据简介

Market Maker(做市商)

- 假设有一天，订单簿变成如下图所示，可以发现，当前市场是缺少流动性的。买一和卖一总有一定的价差，此时交易无法进行。做市商的主要职责是为市场提供流动性，他们通常会双边报价，因为做市商的存在，市场会变得更加有流动性。

bid #	price	ask #
	151	20
	150	12
	149	1
	148	
5	147	
2	146	
	145	
16	144	

■ 常用特征概念

Bid/Ask Spread:

$$BidAskSpread = \frac{BestAsk}{BestBid} - 1$$

最优卖价和最优买价的差除以最优买价

Weighted Average Price:

$$WAP = \frac{BidPrice_1 * AskSize_1 + AskPrice_1 * BidSize_1}{BidSize_1 + AskSize_1}$$

WAP(加权平均价)，Kaggle比赛基于WAP计算已实现波动率(Realized Volatility)。

（WAP的计算不使用BidPrice*BidSize，原因在于作者认为这种定义更能够反映市场价格，举例来说：如果买单远大于卖单（bidsize>>asksize）价格应该上涨，WAP价格应该更接近卖价。）

对数收益率：

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

对数收益率一个优点 $r_{t_1, t_2} + r_{t_2, t_3} = r_{t_1, t_3}$

波动率

Realized Volatility(已实现波动率):

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

Implied Volatility(隐含波动率):

B-S公式（欧式看涨和看跌期权）：

$$c = S_0 N(d_1) - Ke^{-rT} N(d_2)$$

$$p = Ke^{-rT} N(-d_2) - S_0 N(-d_1)$$

$$d_1 = \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma \sqrt{T}}$$

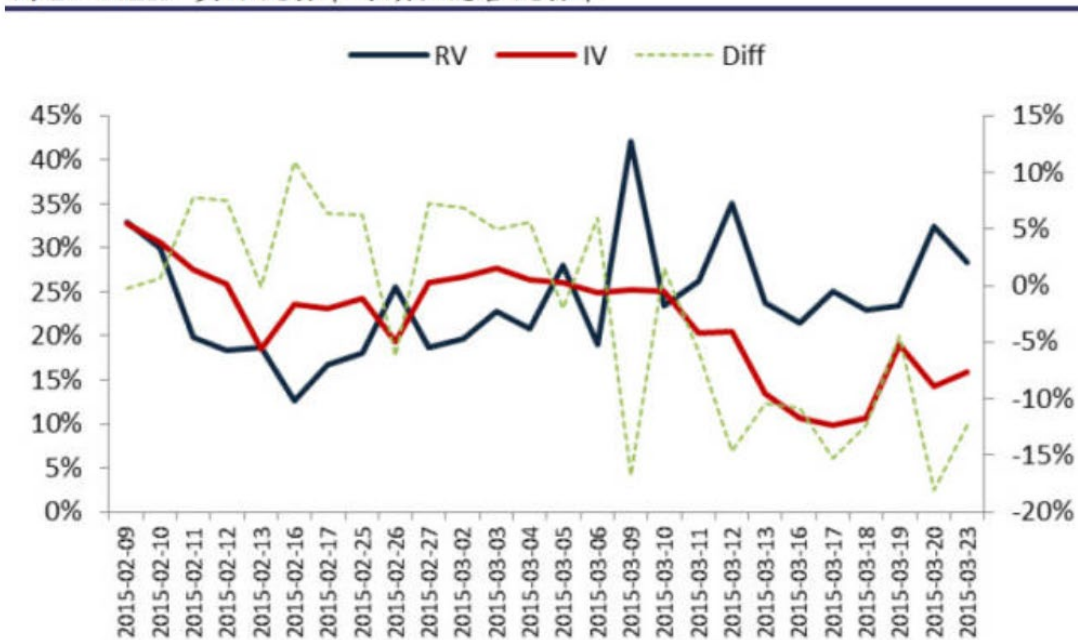
$$d_2 = \frac{\ln(S_0/K) + (r - \sigma^2/2)T}{\sigma \sqrt{T}} = d_1 - \sigma \sqrt{T}$$

根据B-S公式，给定期权的价格，可以反推波动率 σ ，即隐含波动率，该波动率与期权价格一一对应。

为什么要预测已实现波动率

1 利用已实现波动率与隐含波动率的关系进行波动率套利

图 2、50ETF 实际波动率与期权隐含波动率



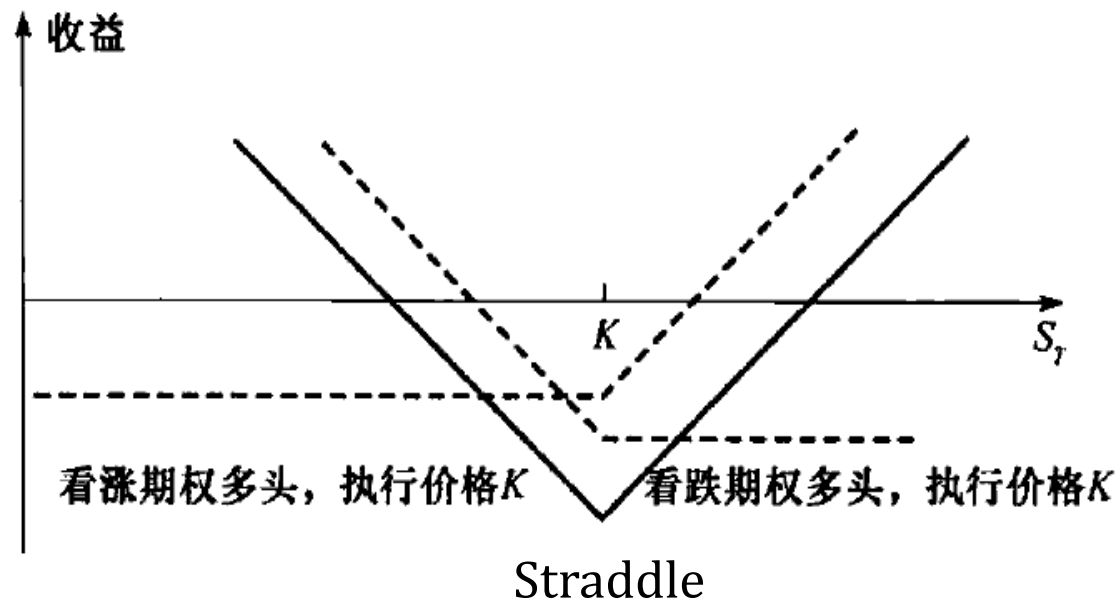
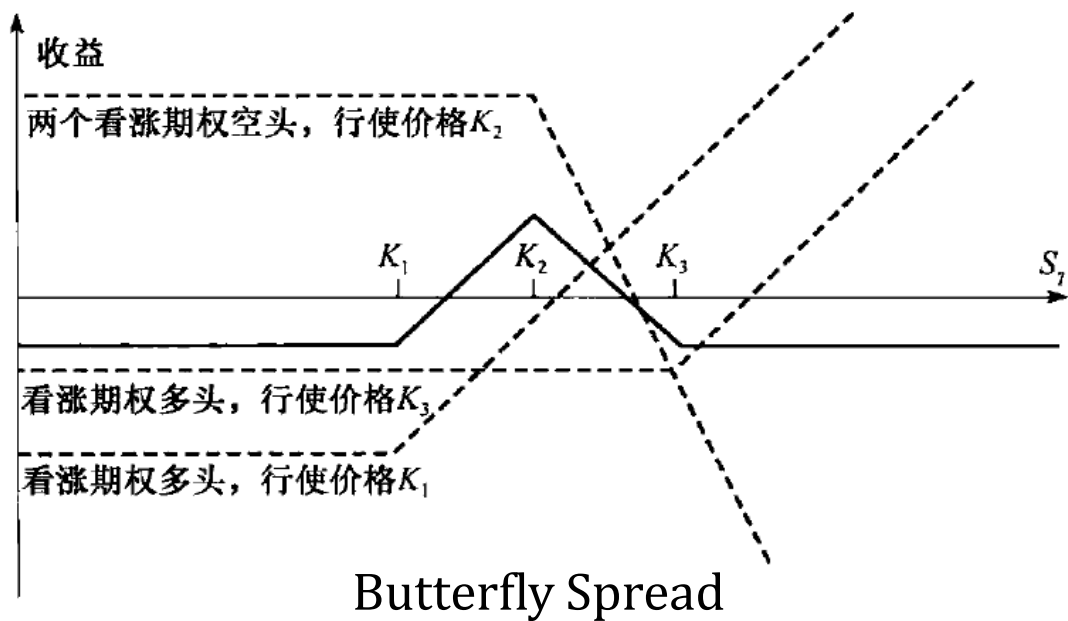
资料来源：兴业证券研究所

原理：已实现波动率与隐含波动率高度相关，隐含波动率又和期权价格存在一一对应。

策略：当预测的已实现波动率高于隐含波动率时买入期权，当预测的已实现波动率低于隐含波动率时卖出期权。同时在标的市场交易标的资产保持 Delta 中性

为什么要预测已实现波动率

2 利用Straddle或Butterfly Spread做空或做多波动率（交易标的保持组合Delta中性）



3 波动率互换



比赛数据介绍

比赛要求以10min长度的窗口，计算已实现波动率，并预测下一个10min窗口的已实现波动率。

Book

	time_id	seconds_in_bucket	bid_price1	ask_price1	bid_price2	ask_price2	bid_size1	ask_size1	bid_size2	ask_size2	stock_id
0	5	0	1.001422	1.002301	1.00137	1.002353	3	226	2	100	0
1	5	1	1.001422	1.002301	1.00137	1.002353	3	100	2	100	0
2	5	5	1.001422	1.002301	1.00137	1.002405	3	100	2	100	0
3	5	6	1.001422	1.002301	1.00137	1.002405	3	126	2	100	0
4	5	7	1.001422	1.002301	1.00137	1.002405	3	126	2	100	0

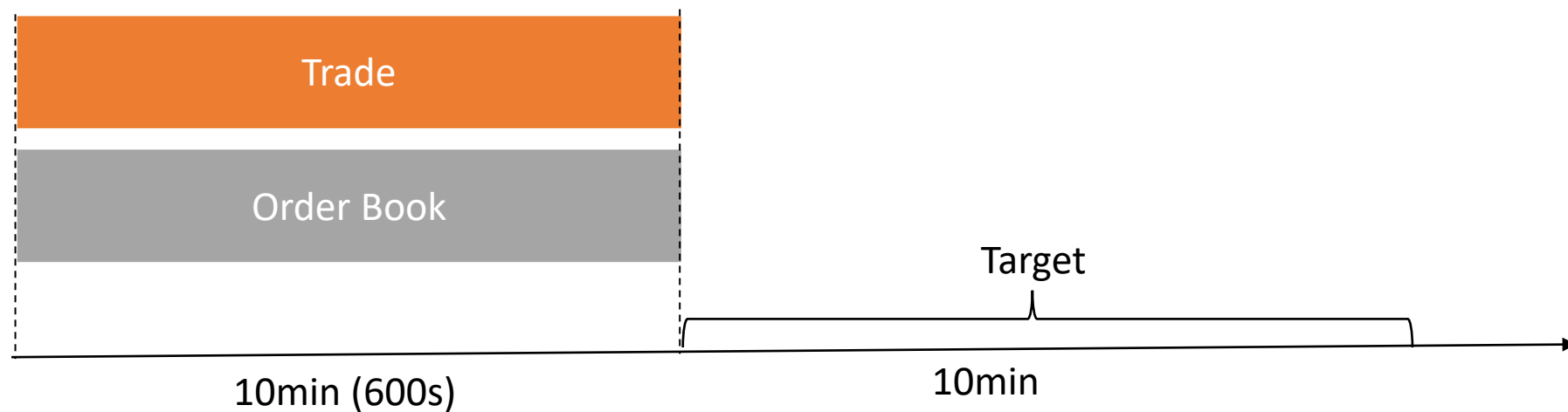
Trade

	time_id	seconds_in_bucket	price	size	order_count	stock_id
0	5	21	1.002301	326	12	0
1	5	46	1.002778	128	4	0
2	5	50	1.002818	55	1	0
3	5	57	1.003155	121	5	0
4	5	68	1.003646	4	1	0

Stock_id: 股票ID
Time_id: time bucket (10min 窗口) 的ID
Seconds_in_bucket: 表示Bucket开始之后的秒数（0-599）
Order_count: 当前交易的发单个数
Price: 一秒内发生的已执行交易的平均价格
Size: 交易量



■ 比赛数据介绍



提交数据格式

	stock_id	time_id	target
0	0	5	0.004136
1	0	11	0.001445
2	0	16	0.002168
3	0	31	0.002195
4	0	62	0.001747

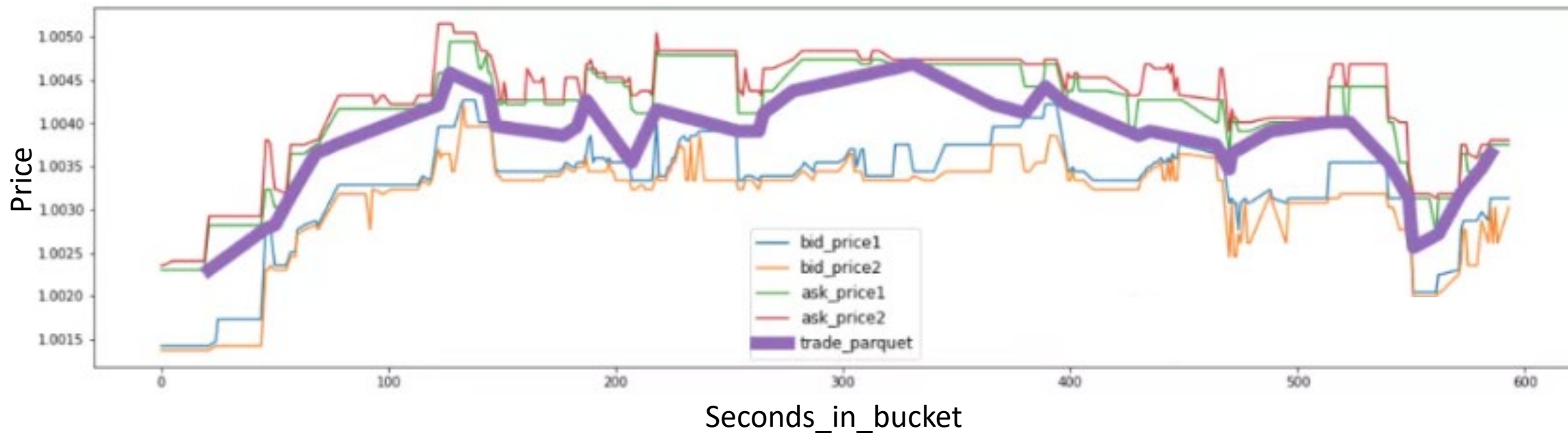


	row_id	target
0	0-4	0.003048
1	0-32	0.003048
2	0-34	0.003048



比赛数据可视化

一支股票在一个window中的订单簿和交易数据



1. 订单簿数据密集程度远大于交易数据
2. 所有交易都是在买一或卖一的价格下进行的（说明这段时间没有发出超过买一或卖一单量的大单）

Kaggle波动率预测比赛专题研究

1 Kaggle比赛简介

2 特征工程

3 模型介绍

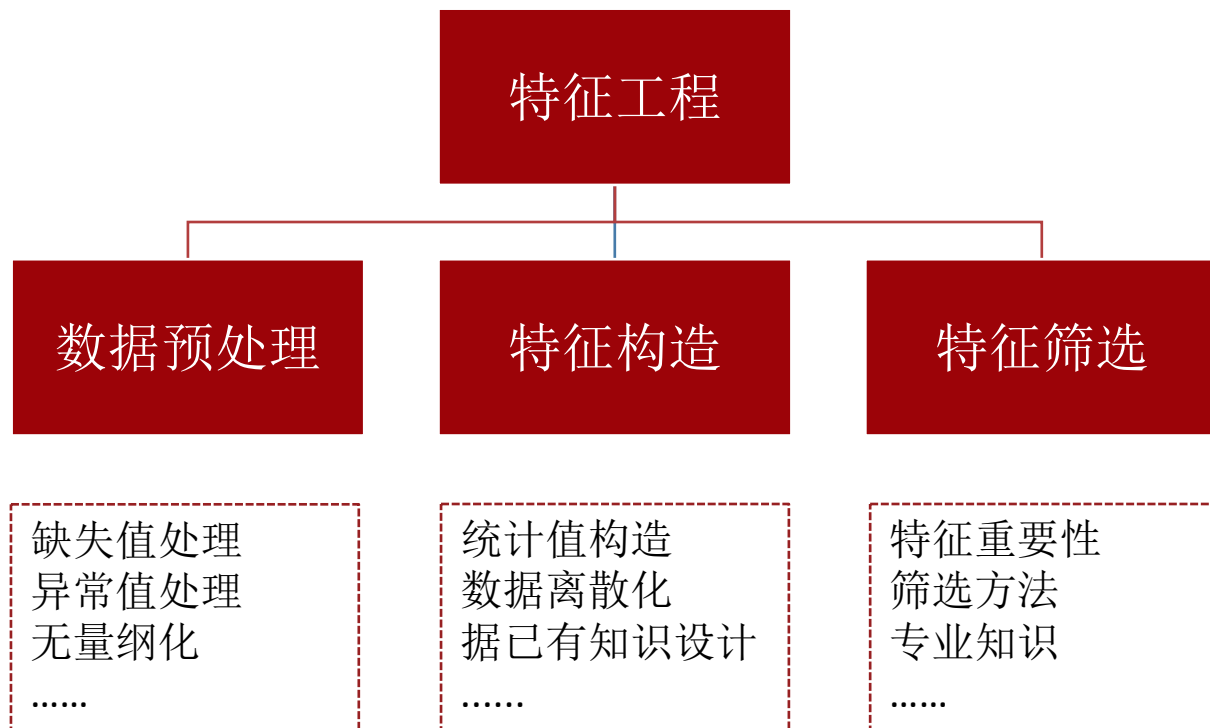
4 PB 2nd方案详解



特征工程

概念

- Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. —Wikipedia
- “数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已”



- 特征构造
 - 1.这个特征是否对目标有实际意义?
 - 2.如果有用，这个特征重要性如何?
 - 3.这个特征的信息是否在其他特征上体现过?
- 特征筛选
 - 1.需要领域知识、经验以及数学知识综合考量特征的有效性，防止胡乱构造
 - 2.在模型迭代验证其是否有正向作用
 - 3.特征选择判定特征重要性



数据预处理

区间放缩

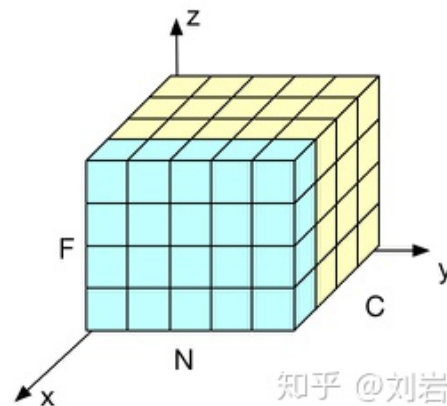
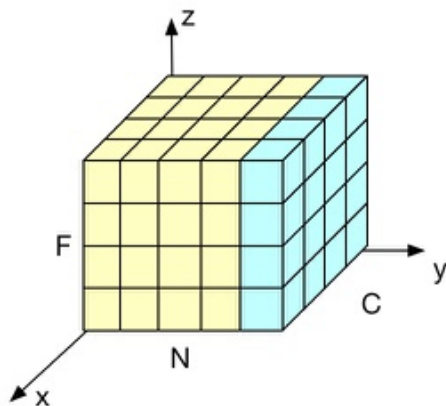
- GaussRank: 保留排序信息, 将数据范围转化至 $[-1,1]$, 生成正态分布
- Power: 指数变换
- Quantile: 分位数变换
- MinmaxScaler: 将数据范围变化到 $[0,1]$
- StandardScaler: 标准化

缩尾处理

- 尾部分布处理: 观察到很多feature的分布都右偏, 因此将top 1%缩尾

变换范围

- 1、对每只股票分别进行标准化的效果比全样本空间变化的效果好
- 2、Layer VS Batch Norm





特征构造-统计值构造

单个变量

- 价格、数量等数据取mean,sum, std, min, max, quantiles等统计值
- 具体做法：
将10分钟分成6个或更多时间窗口，按时间窗口计算feature统计值

多个变量

- 特征按 stock_id 或 time_id指标聚合后，计算mean等统计值
- 具体做法
分别对训练集与验证集聚合指标，再在时间窗口内对数据取统计值

函数处理

- 对指标在时间窗口内的值取rank处理
- 使用log函数处理偏度大的数据



特征构造-数据离散

N近邻法聚合特征

- **time_id近邻:**
时间序列上，寻找某股票出现类似特征的time_id，并计算特征的平均值（如相似价格、相似波动性）
- **stock_id近邻**
寻找同一时间段有类似属性的股票，并计算特征平均值作为输入
- **对比:** time_id近邻在同一股票时序数据中挖掘相似的历史表现
stock_id近邻在同一时间挖掘相似股票的共性。

两种聚合方式

time-id nearest neighbor



stock-id nearest neighbor



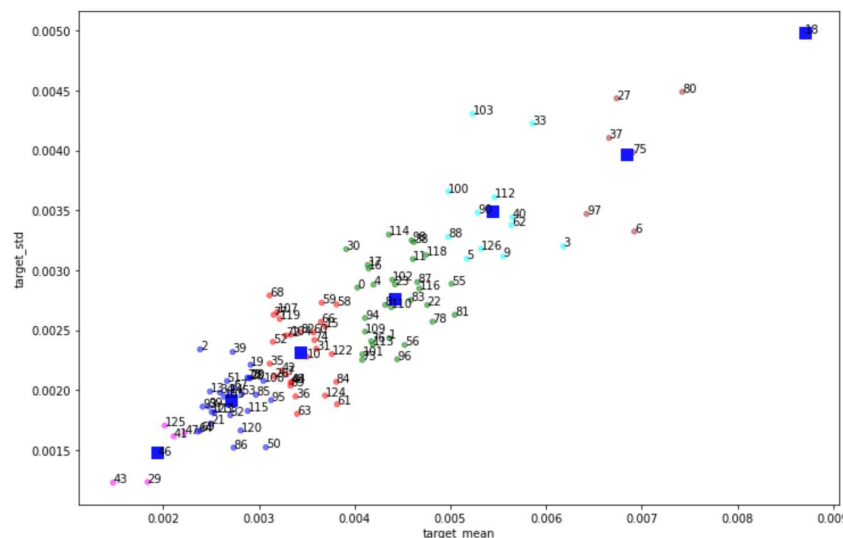


特征构造-数据离散

K-means聚类

- 释义：
用K-means算法聚类进行特征学习找出相似股票，并计算同类特征

- 具体操作：
以target_mean与target_std作为聚类依据，分出n类，并以类中心的均值与方差作为新的特征输入



Identified centroid values:

```
array([[0.00441606, 0.00276504],  
       [0.00684624, 0.00396734],  
       [0.00270886, 0.00192077],  
       [0.00544271, 0.00349715],  
       [0.00343427, 0.00231943],  
       [0.00193308, 0.00147851],  
       [0.00870681, 0.0049812 ]])
```

实践



特征构造-据已有知识

行业知识构造特征

- 构造流动性指标
根据流动性定义结合已有数据构造指标

$$Liquidity_1 = \sum_i \left[\frac{BidSize_i}{Wap_1 - BidPrice_i} + \frac{AskSize_i}{AskSize_i - Wap_1} \right]$$

- 结合流动性与交易量构造TVPL(Trade Volume Per Liquidity)

$$TVPL = \frac{TradeVolume_i}{Liquidity_i}$$

交易量除以流动性与波动率具有显著的相关性：
 $\log(TVPL)$ 与 $\log(vol)$ 的相关性为0.88

- 拓展官方定义的WAP指标

实践

- 优势：特征对预测目标有实际意义
构造有针对性且可能有奇效



特征筛选-参考准则

- **Explainable:** 去除在市场视角下不合理特征，即使其提高了CV和LB
- **实际效果:** 去除不能提高稳定提高CV的特征，即使理论上它应该可以
- **重要性得分: Shapley Values选择法**
 - **优势:** 可以给出一致的特征重要性得分（与树模型比较）
 - **方法:** 分别计算含有/去除某个特征的模型在样本上的拟合效果，作差即可视为该模型的贡献度
- **筛除方法: Recursive Feature Elimination**
 - 将需要筛选的特征放入模型计算各个特征的重要性得分
 - 移除特征重要性最低的一个特征，得到一个新的特征子集再次进行训练
 - 重复上一步直至遍历完所有特征，选择分类精度最高的特征子集作为最优特征组合
- **实例:**
 - 用Shapley feature importances 递归删除特征
 - 对每个fold都进行操作，得到5组特征，数量从40到130不等（每个fold跑了8小时）
 - 保留了出现4次及以上的特征，总共有45个，在public test中达到0.1975



Tricks-目标处理

clipping

- 将预测结果异常 (≤ 0 或 ≥ 1) 的用naïve prediction代替

剔除outliers

- 将数据集里target值异常的视为噪音，进行删除。判断标准： $\text{target/realized volatility} > 3$

目标转换

- 考虑到不同股票不同时间的未来十分钟波动率的绝对变化可能较剧烈，但是相对于当前十分钟的波动率的相对变化可能较小
- 预测未来波动率与当前波动率的比值，而非波动率本身

多模型多种子集成

- 将几个不同模型的结果加权平均，如LGBM+神经网络+naïve prediction
- 单个模型的结果也由几个种子的结果简单平均，如LGBM三个种子的结果平均



Tricks-domain knowledge

特殊情况处理

- 市场交易停止导致某段时间数据缺失
- 某只股票离开数据集：退市或者合并
- 股票拆分：order book的size提升数量级
- 市场非常活跃导致运算需要的内存增加

t-SNE

- 非线性的降维方法，将欧氏距离转化为条件概率来表征点间相似度
- 使用梯度下降算法来使低维分布学习/拟合高维分布
- 用t分布代替标准正态，使得尾部不那么“拥挤”

信息复原

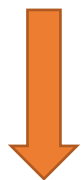
- Price: 考虑到价格的数量级会包含信息，因此将price乘以一个倍率（并不work）
- Price: 考虑到tick size为0.01，因此可以将标准化后的价格通过denormalize复原
- Time: 考虑到时间先后顺序会蕴含信息，因此通过t-SNE降维来恢复时间顺序



Tricks-其他

用线性拟合替代细分窗口

- 将10分钟时间段分成6个或者更多的小窗口
- 每个小窗口计算数字特征等feature
- 将所有feature输入模型进行训练



- 在整个时间窗口中，用feature值对时间作线性回归
- 用截距、斜率、残差等作为feature，可以替代一些简单的数字特征

变化target范围以利用更多数据

- 将训练集**测试集**合并，并从时间中点断开分为前五分钟数据和后五分钟数据
- 拟合5分钟预测模型，即0~5分钟的feature预测6~10分钟的target
- 使用上述模型，基于6~10分钟feature预测11~15分钟的target
- 将上述预测值作为feature输入到主模型

提升速度

- Rapids Cudf类似于pandas，但是其在GPU中运行，要比后者快很多

Kaggle波动率预测比赛专题研究

1 Kaggle比赛简介

2 特征工程

3 模型介绍

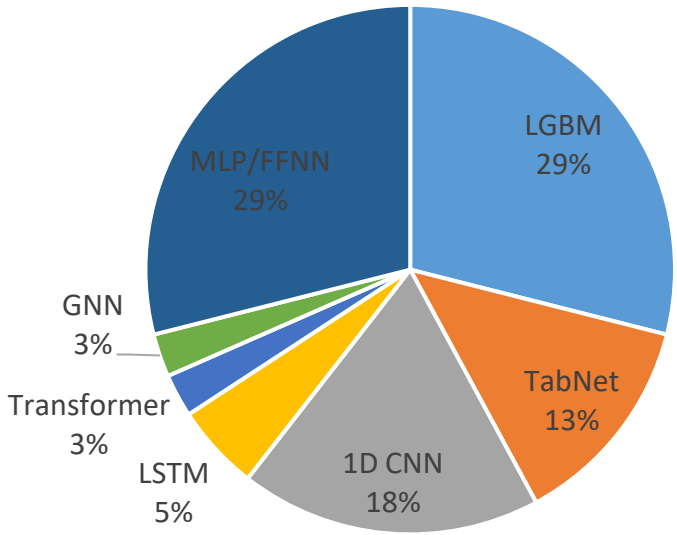
4 PB 2nd方案详解

比赛模型统计

常用模型

- TabNet: 专门针对表格型数据设计的网络结构
- Transformer: 利用Attention机制形成的Encoder-Decoder结构
- LGBM: Light Gradient Boosting Machine 轻量级梯度提升机
- 1D CNN: Convolutional Neural Network 一维卷积神经网络
- LSTM: Long-Short Term Memory 长短期记忆人工神经网络
- GNN: Graph Neural Network 图神经网络
- MLP: Multilayer Perceptron 多层感知机,即FFNN(Feed-Forward Neural Network) 前馈神经网络

已公开的PB top100方案模型使用次数分布

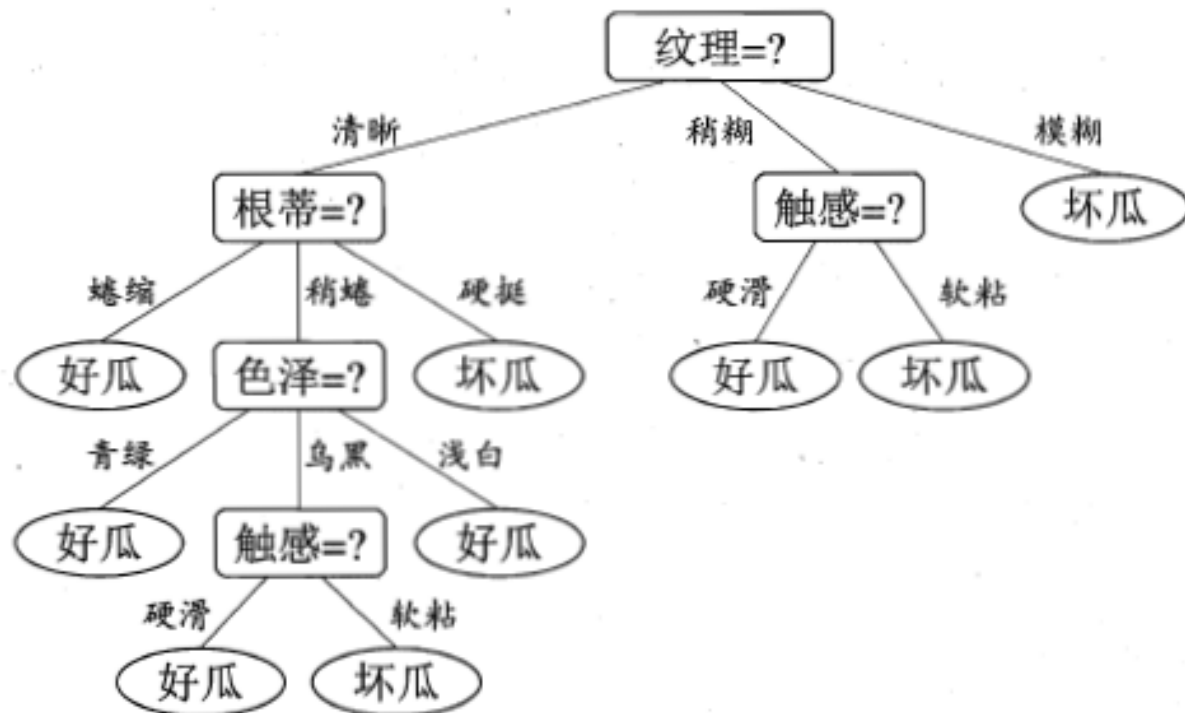


模型方法	使用次数
LGBM	11
TabNet	5
1D CNN	7
LSTM	2
Transformer	1
GNN	1
MLP/FFNN	11

数据根据比赛网页code和discussion部分统计整理

决策树

决策树是一种树形结构，每个内部节点表示一个属性上的判断，每个分支代表一个判断结果的输出，最后每个叶节点代表一种分类结果。



Reference: <https://zhuanlan.zhihu.com/p/128472955>

常见类型

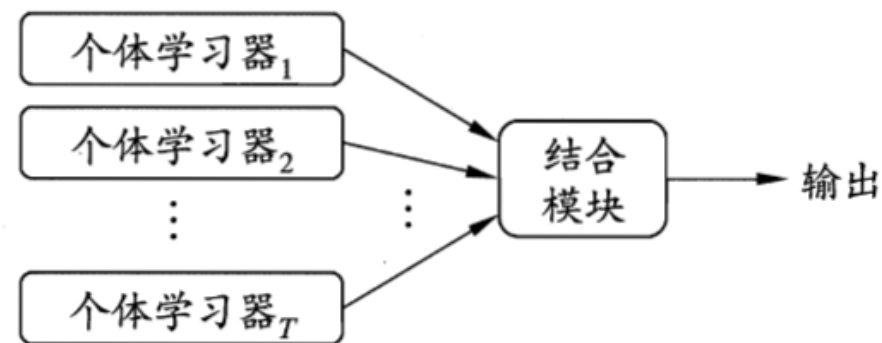
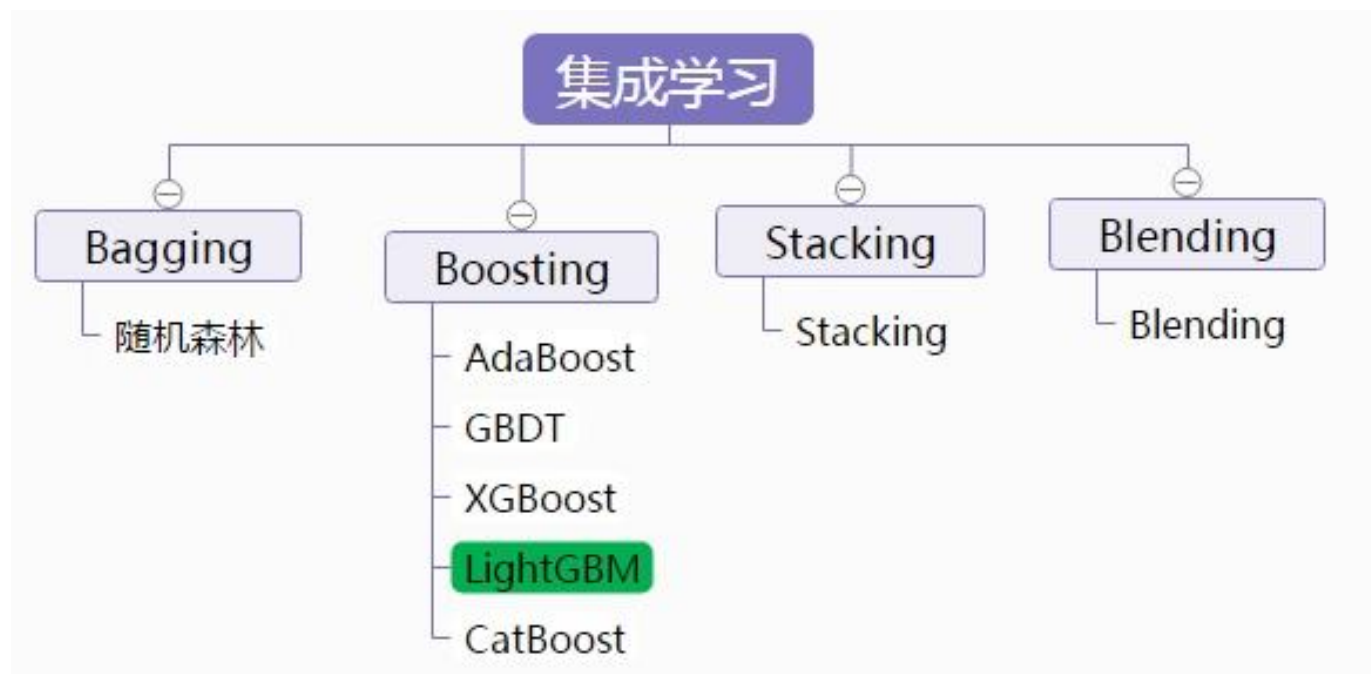
- **ID3**: 使用信息增益作为特征选择的度量
- 熵的不断最小化，实际上就是提高分类正确率的过程
- 存在的问题：越细小的分割分类错误率越小

$$\text{Ent}(\tilde{D}) = - \sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k$$

- **C4.5**: 使用信息增益比作为特征选择的度量
- **CART**: （分类回归树）
- 不仅可以做分类，还可以做回归（二叉树）
- 分类树：基尼指数
- 回归树：样本的最小方差

Ensemble (集成学习)

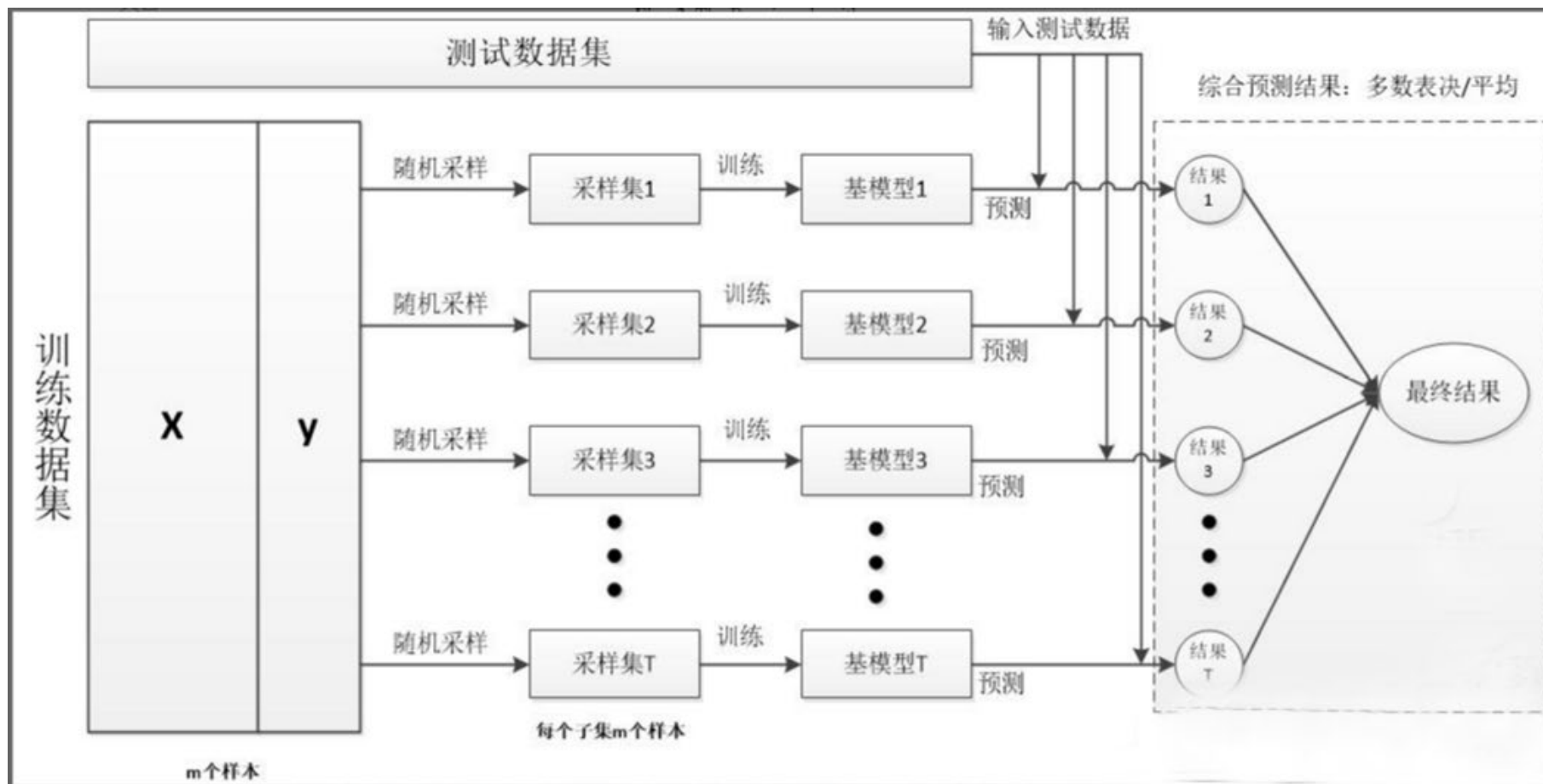
集成学习是构建多个学习器并通过一定的策略结合成一个性能更佳的学习器。



Reference: 《机器学习》 周志华（西瓜书）第八章 集成学习

Bagging

从训练集中进行子抽样组成每个基模型所需要的子训练集，对所有基模型预测的结果进行综合产生最终的预测结果。

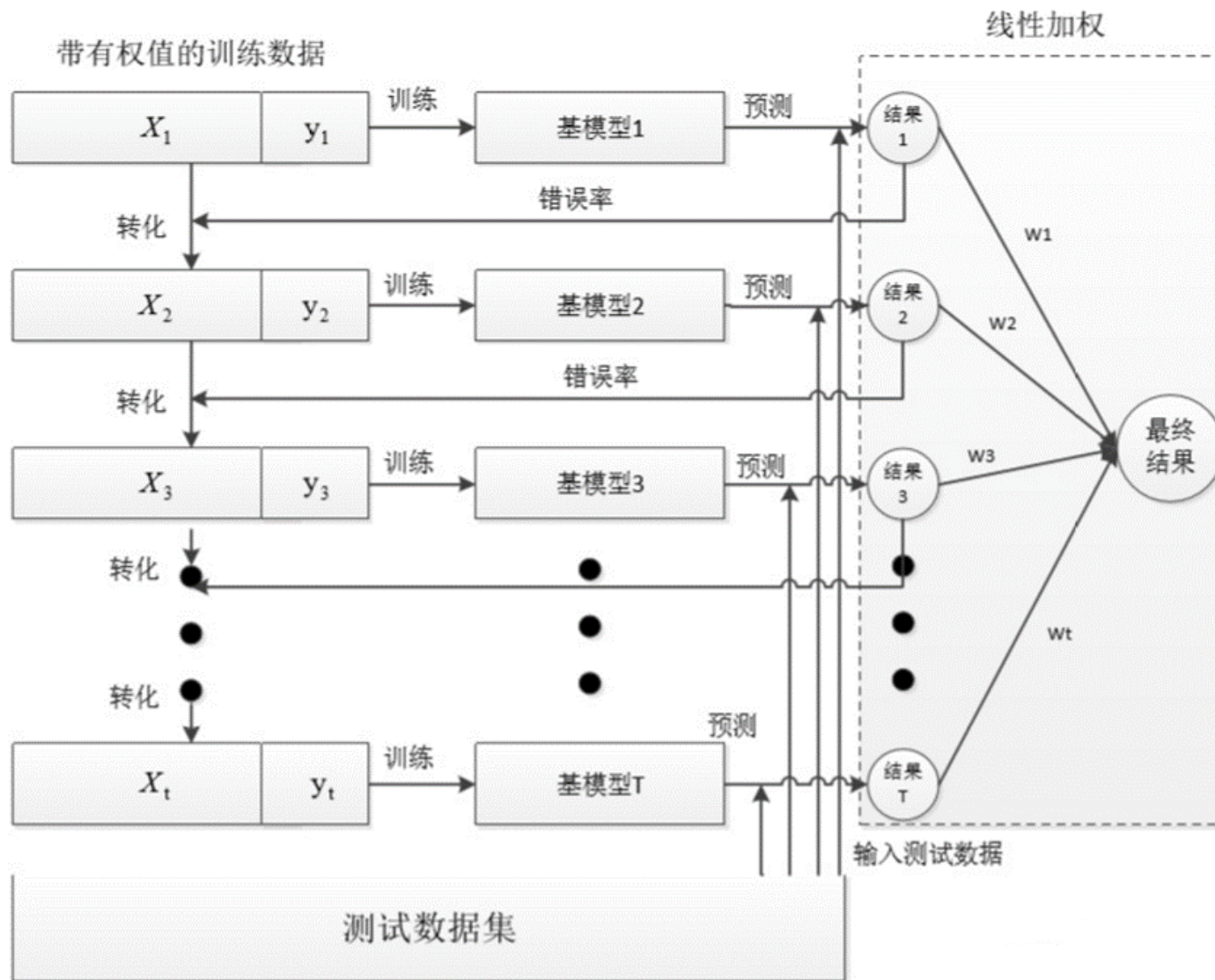


步骤

- 第一步：在训练数据集中随机采样
- 第二步：对不同的子集进行训练，得到基模型
- 第三步：对基模型分别进行预测
- 第四步：将结果综合
- 分类任务：投票
- 回归任务：平均

Boosting

训练过程为阶梯状，基模型按次序一一进行训练，基模型的训练集按照某种策略每次都进行一定的转化，对所有基模型预测的结果进行线性综合产生最终的预测结果。

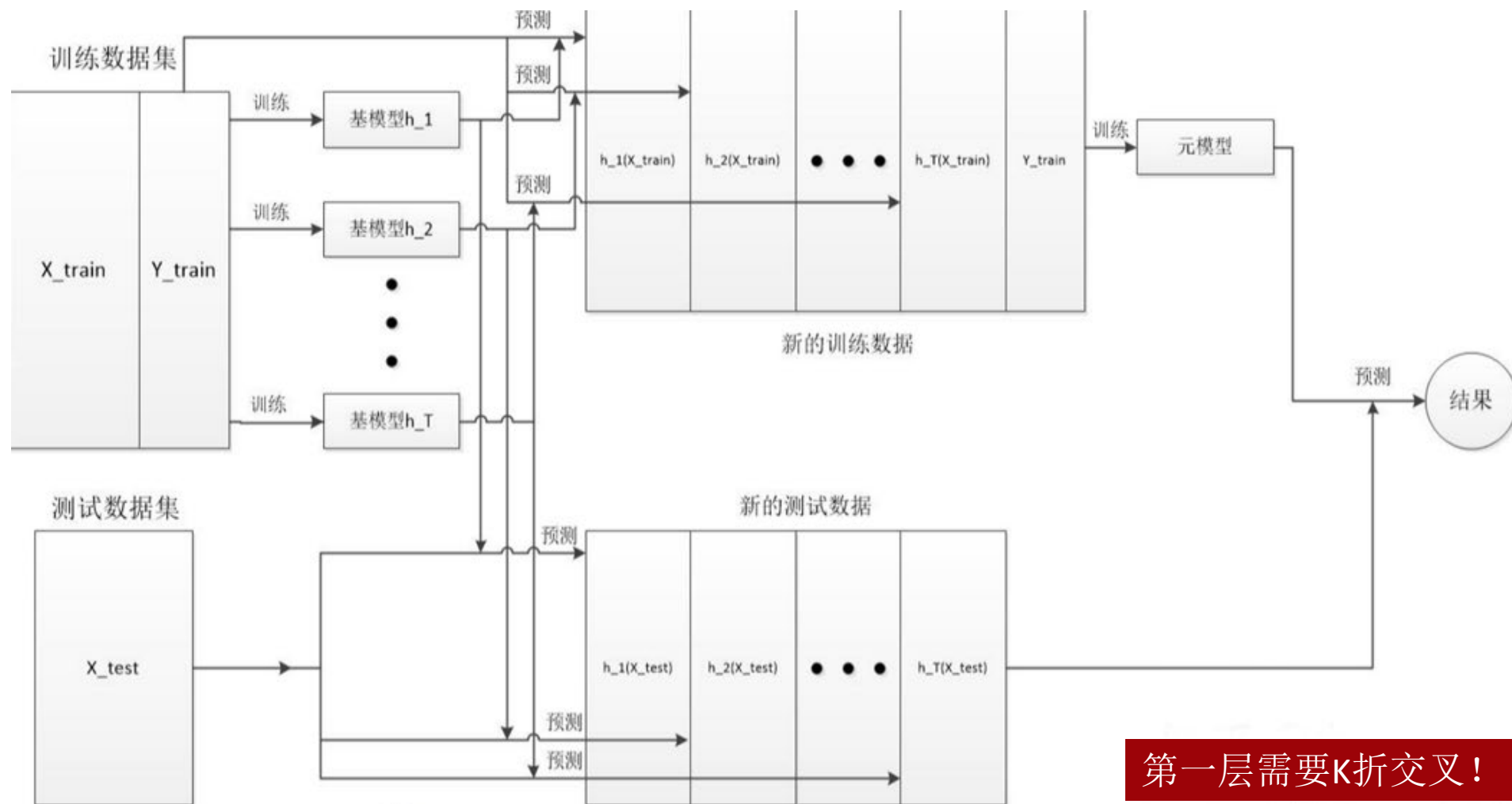


步骤

- 第一步：初始化训练数据的权重
- 第二步：训练第一个基模型，计算错误率，计算模型的系数
- 第三步：更新数据集的权重，误分类数据的权重调大，分类正确的数据权值调小，再训练一个基类模型
- 第四步：每个模型对测试集进行预测
- 第五步：对所有基模型的预测结果进行加权求和

Stacking

新的训练集和新的测试集



步骤

- 第一步：训练不同的基模型
- 第二步：使用基模型，分别对训练数据进行预测，与原始训练数据的标签一起组成新的训练数据
- 第三步：使用基模型，分别对测试数据进行预测，生成新的测试数据
- 第四步：使用新的训练数据，训练一个元模型
- 第五步：使用元模型对新的测试数据进行预测，得到最终结果

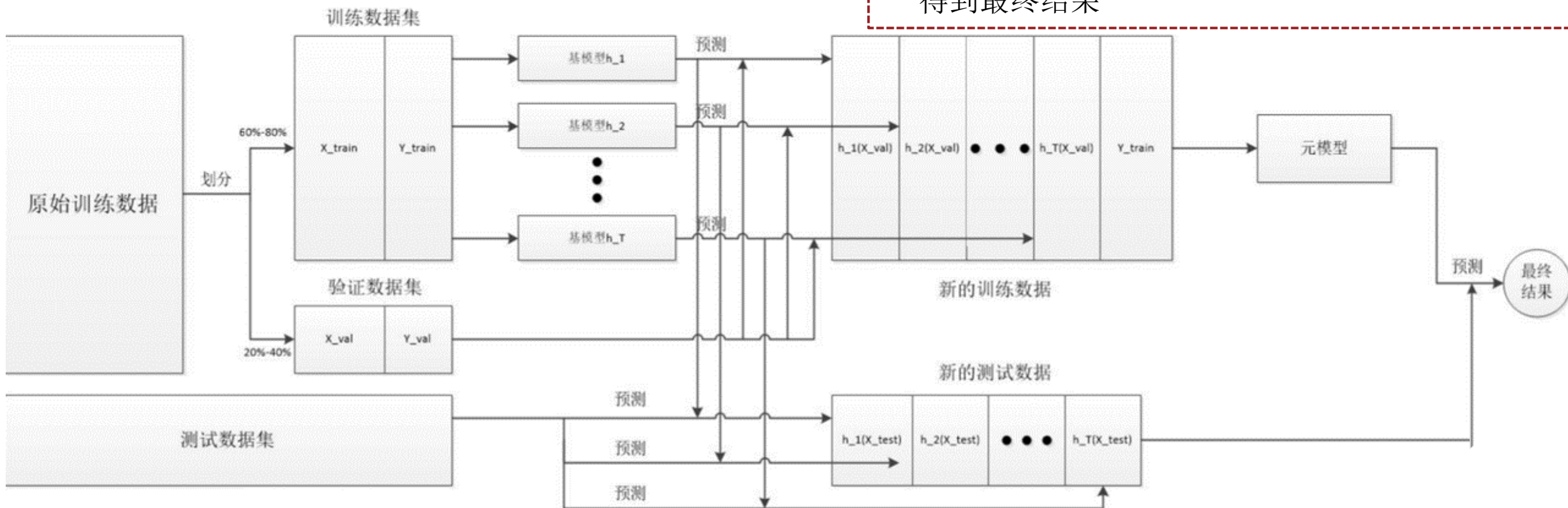
第一层需要K折交叉！！（过拟合）

Blending

数据划分为训练集和验证集+
新的训练集和新的测试集

步骤

- 第一步：将原始训练数据划分为训练集和验证集。
- 第二步：训练不同的基模型
- 第三步：使用基模型，对验证集进行预测，结果作为新的训练数据
- 第四步：使用新的训练数据，训练一个元模型
- 第五步：使用基模型，对测试数据进行预测，结果作为新的测试数据
- 第六步：使用元模型对新的测试数据进行预测，得到最终结果



集成学习算法比较

串并行结构				
	样本选择	样例权重	预测函数	并行计算
Boosting	不变	错误率越高权重越大	分类误差率越低权重越大	不可以并行
Bagging	随机采样选取	权重相等	权重相等	可以并行

分层结构				
	数据划分	稳健型	时间	数据使用
Stacking	K-Fold CV	较好	较长	存在数据泄露
Blending	Hold-Out	较差	较短	完全分开

Reference: <https://zhuanlan.zhihu.com/p/126968534>

Bias-Variance

准确是两个概念。准是 bias 小，确是 variance 小。

偏差 vs 方差

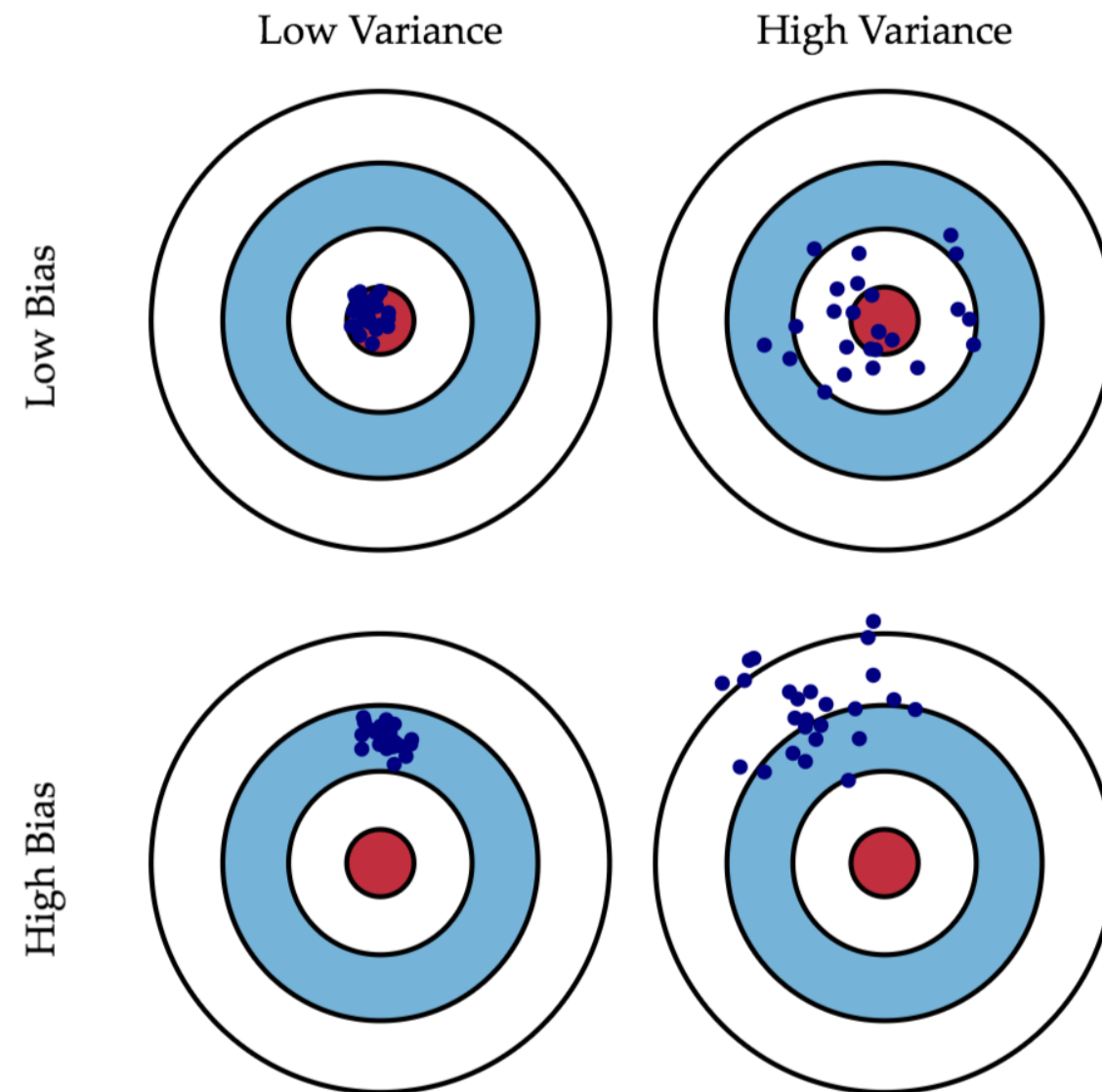
偏差：学习拟合出来结果的期望与真实规律之间的差距

$$\text{Bias}(X) = E[\hat{f}(X)] - f(X)$$

方差：学习拟合出来的结果自身的不稳定性

$$\text{Var}(X) = E[(\hat{f}(X) - E[\hat{f}(X)])^2]$$

$$\text{Err}(X) = \text{Bias}^2 + \text{Variance} + \sigma_{\varepsilon}^2$$



Reference: <https://liam.page/2017/03/25/bias-variance-tradeoff/>

Boosting vs Bagging

偏差 vs 方差

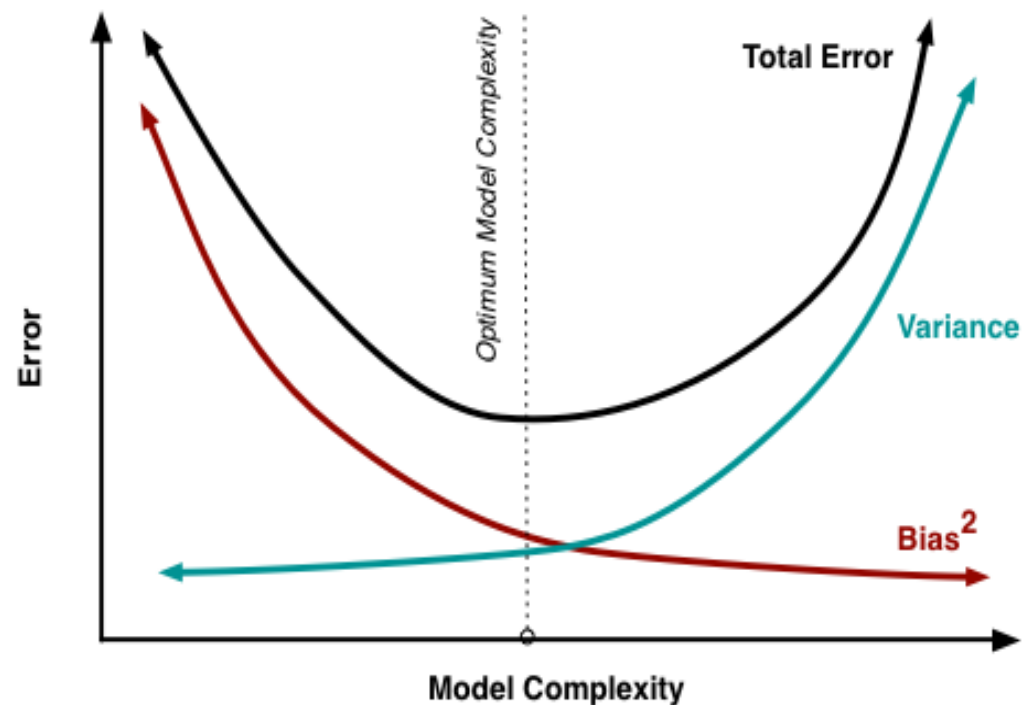
Boosting方法通过逐步聚焦分类器分错的样本，sequential地最小化损失函数，**减少偏差**；但是子模型强相关，不能显著降低方差。

Bagging采用分而治之的策略，通过对样本多次采样，分别训练多个模型，**减少方差**；但是bagging后的偏差和单个子模型接近，所以不能显著降低偏差。

适用类型

Boosting主要关注降低偏差，因此Boosting能基于泛化性能相当弱的学习器构建出很强的集成；Bagging主要关注降低方差，因此它在不剪枝的决策树、神经网络等学习器上效用更为明显。

$$\frac{\text{Var}(X_i)}{n} \leq \text{Var}\left(\frac{\sum X_i}{n}\right) \leq \text{Var}(X_i)$$



CART(Classification And Regression Tree)

- 处理样本输出为连续值的情形。使用MSE作为损失函数，而非基尼系数
- 伪代码

def **chooseBestSplit**(dataSet, ops) #用于寻找最佳特征与最佳切分点，ops是退出机制的上下限

for every feature:

part1, part2 = split data set based on the feature value

if part1 or part2 is small: #切分出的数据集太小

continue

new cost = MSE(part1)+MSE(part2)

if change of cost is too less, break

update the cost, save the temp feature and value

return best feature, best split value

def **CreateTree**(dataSet, ops): #用递归方法建立回归树

best feature, best split value = chooseBestSplit

generate the left tree and right tree through recursion

特征选择：计算各个特征各个值划分的两部分 D_1 与 D_2 的误差平方和，选择误差平方和最小的作为最优特征和最优切分点

条件判断：根据最优特征与特征点构建 D_1 与 D_2 ，重复上述步骤直至符合退出条件

CART

```
def prune(tree, testData): #剪枝
```

```
    find the lTree and rTree
```

```
#对左子树与右子树递归
```

```
    if lTree and rTree not exist:
```

```
        return number
```

```
    if lTree not a leaf:
```

```
        prune(lTree)
```

```
    if rTree not a leaf:
```

```
        prune(rTree)
```

```
    if both are leaf: #当左右均为节点时，判断合并两节点能否有更小的MSE
```

```
        calculate the MSE of emerging the two sub trees together
```

```
        compare the MSE to decide whether to emerge
```

```
        return Tree
```

```
    else:
```

```
        return new Tree
```

容易造成决策树划分得太细，会对数据产生过拟合，因此要通过剪枝来解决。通过递归判断哪些部分需要被合并（剪枝）

GBDT(Gradient Boosting Decision Tree)

- 将CART按照boosting的方法连接起来，就构成了GBDT；把CART用bagging的方式组合起来，那就是随机森林
- 追加法训练：训练好的不再调整，只追加新的训练子系统
- 考虑损失函数： $L(y, f(x)) = L(y, y^{t-1} + f_t(x))$, 其中 $f(x) = \sum_1^N f_i(x)$ ，表示 N 个子学习分类器的预测之和。
 $f_t(x)$ 为新加入的CART的预测结果，目标是使其最小。

考虑 L 本身为平方损失函数的情形，为求导方便乘一个 $\frac{1}{2}$ ，即：

$$L(y, f(x)) = \frac{1}{2} (y_i - F(x_i))^2$$

对上式求导求极值有：

$$\frac{\partial L}{\partial F} = F(x_i) - y_i = \text{gradient}$$

即残差为梯度的相反数。因此我们只需要对残差回归拟合即可。具体做法即：每次用新生成的CART与标签比较计算残差，然后用残差作为新标签值拟合下一个CART。同时为了防止过拟合，可以通过人为设置层数惩罚项等方式进行控制。不如在XGBoost模型中其选用正则化项来控制模型复杂度。

■ GBDT优缺点

➤ 优点

- ①预测阶段的计算速度快，树与树之间可并行化计算。
- ②在分布稠密的数据集上，泛化能力和表达能力都很好。
- ③采用决策树作为弱分类器使得GBDT模型具有较好的解释性和鲁棒性，能够自动发现特征间的高阶关系，并且也不需要数据特殊的预处理如归一化等。

➤ 缺点

- ①GBDT在高维稀疏的数据集上，表现不如支持向量机或者神经网络。
正则项约束由于只考虑了树本身的深度与叶节点的权重，对于稀疏数据集基本没用，极易过拟合
- ②GBDT在处理文本分类特征问题上，相对其他模型的优势不如它在处理数值特征时明显。
- ③训练过程需要串行训练，只能在决策树内部采用一些局部并行的手段提高训练速度。

XGBoost

- 定义树的复杂度函数为 $\Omega f(t) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T \omega_j^2$; 其中 T 为叶子节点数量
- 使用的是二次项的近似, 相较于GBDT采用的一次项近似更为精准。

$$L(y_i, y_i^t) = L(y_i, y_i^{t-1} + f_t(x)) + \Omega f(t) \approx L(y_i, y_i^{t-1}) + g_i f_t(x) + \frac{1}{2} h_i f_t^2(x) + \Omega f(t)$$

- 在基分类器的选择上, GBDT更倾向于采用CART作为分类器, 而XGBoost支持多种类型的基分类器
- XGBoost采用了正则化、Shrinkage、数据采样等多种方式不仅大幅降低了过拟合, 还减少了计算。
- 决策树中最耗时的一个步骤就是对特征的值进行排序, XGBoost在训练之前, 预先对数据进行了排序, 然后保存为block结构, 后面的迭代中重复地使用这个结构, 大大减小计算量。这个block结构也使得并行成为了可能, 在进行节点的分裂时, 需要计算每个特征的增益, 最终选增益最大的那个特征去做分裂, 那么各个特征的增益计算就可以开多线程进行。
- 传统的GBDT没有设计对缺失值进行处理, XGBoost可以自动学习出它的分裂方向。XGBoost对于缺失值能预先学习一个默认分裂方向。

LightGBM

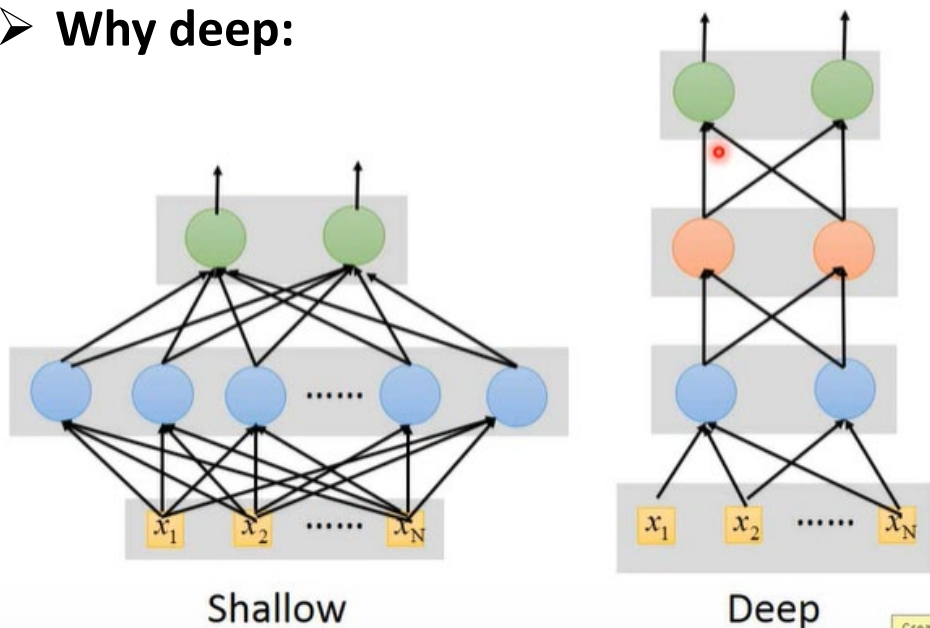
- XGBoost使用的是预排序的方法节省的节点分裂时间开销，但这种做法是要通过牺牲空间换取的。XGBoost除了需要对节点信息的信息进行存储外还需要存每个数据的索引。因此消耗空间大致上是整体数据量的两倍。而LGBM是采用的基于直方图的排序方法，在数据存储上存储的是bin类型的数据，在数据消耗上大约只是XGBoost的1/8。
- 在时间消耗上，LGBM采用的是对数据进行离散化分组，将原有浮点数离散为k个整数，然后再选取每组中的分割点进行分类，时间复杂度较XGBoost的 $O(\text{data} * \text{feature})$ 降低至了 $O(k * \text{feature})$

Why GBDT?

- 模型需要解决的两大问题：训练集的表现（**bias**）与泛用性（**variance**），由于模型学习只能尽可能接近理论中的“完美模型”，因此我们需要在**variance**和**bias**之间进行权衡。
- 模型的训练过程中我们是在用有限的数据（训练集）去匹配无限的数据信息，因此我们总能通过增加参数（加入新的特征、加入新的子学习系统等等）来逼近训练集的“完美模型”。但**GBT**模型相较于传统**LR**模型更具备可控性。**GBT**模型由于是通过将若干个弱学习模型结合得到的最终模型，相较于**LR**模型在参数的调整对整个模型都会有较大的影响，每一次的弱学习模型在**bias**的小幅提升并不会大幅影响模型的泛化能力。
- 树算法整体对噪声数据的抗干扰能力更强。同时在现实情境中多特征数据的分类问题上树模型由于自身在特征选择上的自适应性，往往更能胜任该类任务。
- 树模型相较于**LR**具有灵活、深度的定制功能，在参数选择与损失函数、基分类器选择上均更为灵活。系统具有可扩展性, 可以从容处理更大的数据。

深度学习

- **What:** 深度学习是一种特殊的机器学习。通过建立、模拟人脑进行分析学习的神经网络，并模仿人脑的机制来解释数据的一种机器学习技术。
- **Why learning:** 深度学习无需大量的特征工程即可自行抽取样本数据中的特征信息，模型的每一层结构会根据输入自行提取所需的特征信息用于学习。这也使得深度学习往往在大样本数据输入时表现比ML好，因为模型能够学习到更深层次的特征。同时深度学习能够学习到一些极为复杂的模型，通过加入层数可以较好的提取更多的特征并提升模型的判断能力。
- **Why deep:**



Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*.
Created with EverCam
<http://www.camdemy.com>

深度学习 vs 传统机器学习

深度学习的重要特征

- 层级表示学习(hierarchical representations learning)
 - 赋予深度学习自己挖掘特征的能力
 - 可以把DNN看成是多级知识蒸馏的过程：信息穿过连续的过滤器，其纯度越来越高（即对任务的帮助越来越大）
- 所有参数共同学习(jointly learning)

这两个简单的特性，在数据足够多的情况下能带来魔法般的学习能力。

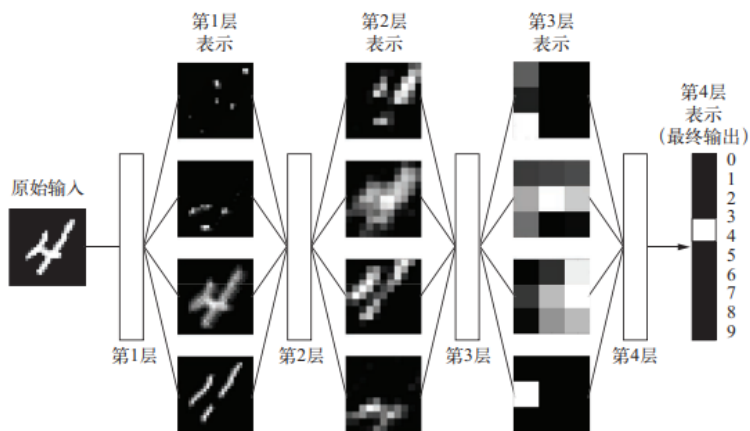


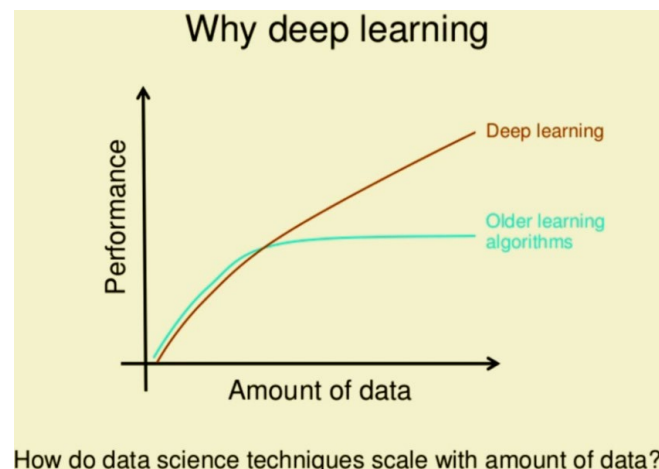
图 1-6 数字图像分类模型学到的深度表示

深度学习的几何解释



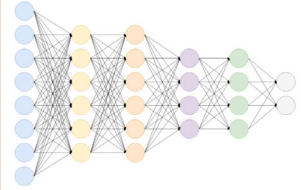
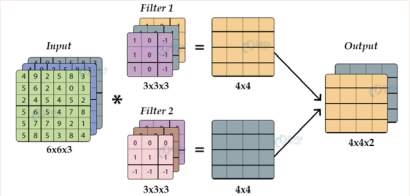
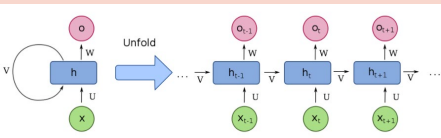
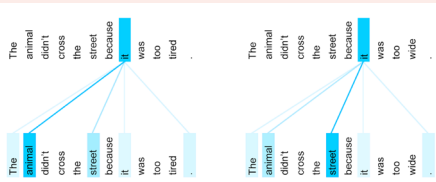
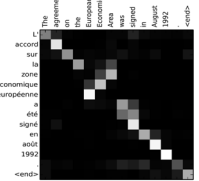
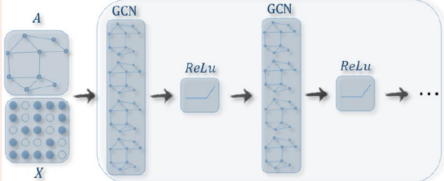
图 2-9 解开复杂的数据流形

让纸球恢复平整就是机器学习的内容：为复杂的、高度折叠的数据流形找到简洁的表示。现在你应该能够很好地理解，为什么深度学习特别擅长这一点：它将复杂的几何变换逐步分解为一长串基本的几何变换，这与人类展开纸球所采取的策略大致相同。深度网络的每一层都通过变换使数据解开一点点——许多层堆叠在一起，可以实现非常复杂的解开过程。



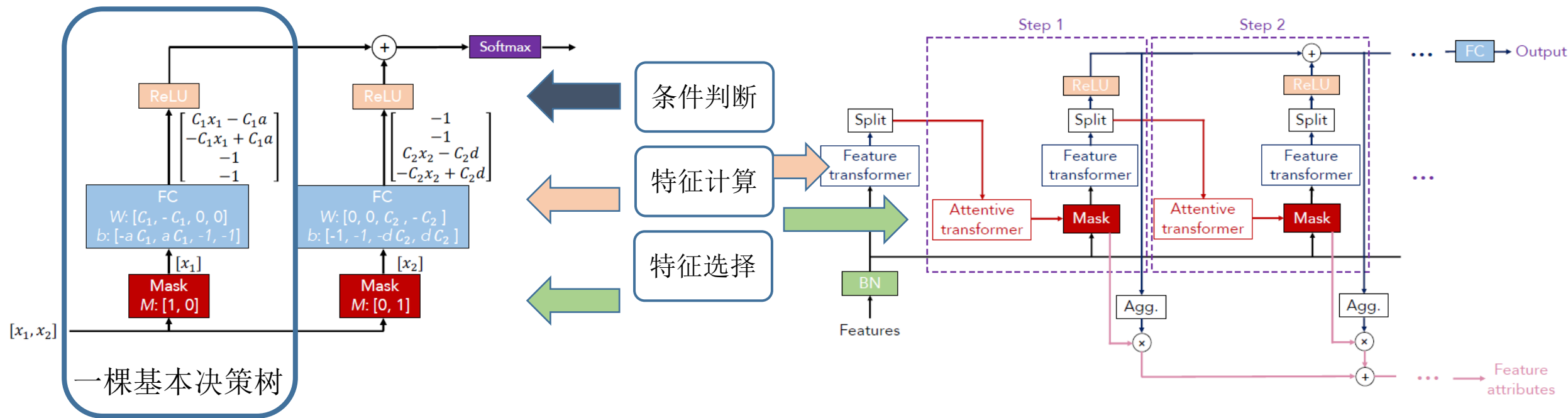


常见神经网络结构介绍

名称	示意图	特点	适用场景	备注
全连接神经网络(FCNN)		无任何归纳偏置 (inductive bias)	通用非线性变换； 对输入作维度变化	
卷积神经网络(CNN)		通过参数共享实现 平移不变性	欲学习的patterns远小于所输入的特征的尺度，且patterns可能存在于不同的位置，降采样不严重影响重要信息	处理时间序列常用一维卷积(1D Conv)
循环神经网络(RNN)		用共享参数的网络结构依顺序处理序列输入	输入序列的位置信息对欲处理的问题重要	对过长的序列处理乏力，容易发生梯度消失和梯度爆炸，目前常用的RNN变体LSTM、GRU一定程度上缓解这一问题，但没有完全解决
自注意力机制(Self-Attention)		模型自己学习如何捕捉上下文的重要信息来更好地表示每一时间步的信息	输入的信息具有长序依赖，即输入的序列中某时间步的信息与相隔遥远的另一时间步的信息具有重要相互作用	和RNN一样都是对序列信息建模的网络结构，相比RNN能适合处理长数据，且不同的时间步能并行运行，时间开销小，但空间开销大。
注意力机制(Attention)		模型自己学习两个序列间的信息对齐	需要模型自己学会序列对齐	欲深入理解attention和self-attention 请参阅神文《 Attention is all you need 》
图神经网络(GNN)		专门处理图数据的神经网络	个体之间的关系对欲处理的问题重要	

Tabnet intuition (模仿决策树进行表格型数据的处理)

- Attention: 本质上是通过对输入数据的加权分配模型对每个数据的关注程度



- Tabnet模型通过Attention transformer进行了Mask的生成，可以理解为当前步骤下模型对输入特征的注意力分布，之前被使用较多的特征会被给予一个较小的Mask值。

Kaggle波动率预测比赛专题研究

1 Kaggle比赛简介

2 特征工程

3 模型介绍

4 PB 2nd方案详解

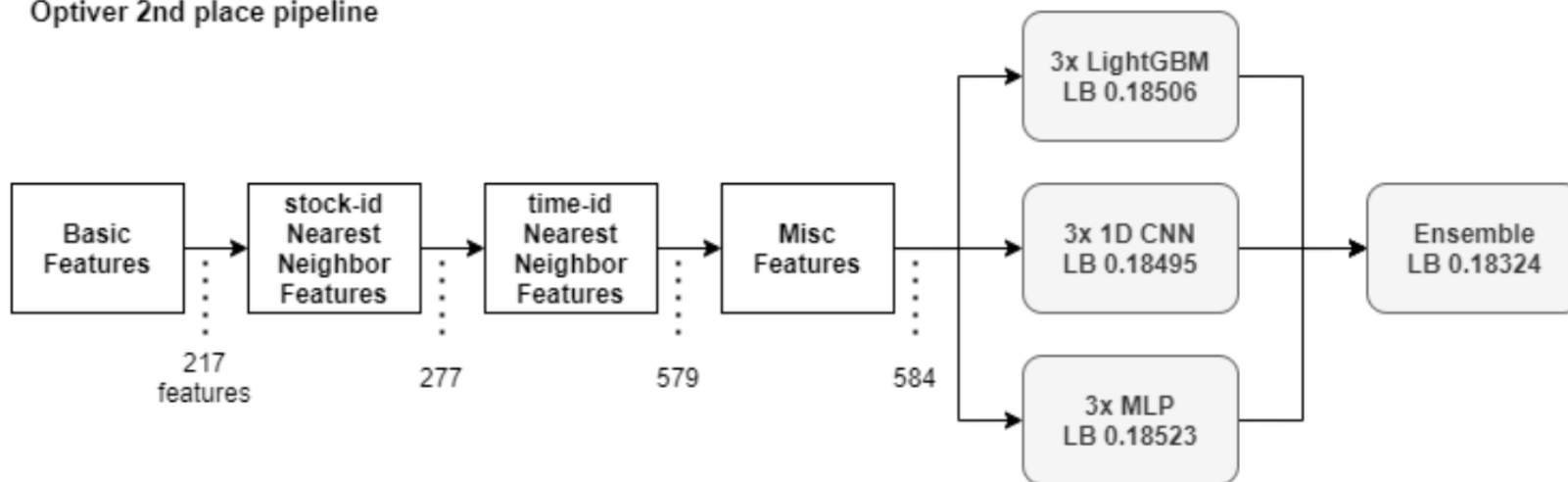


Top2 方案介绍

主要实现思路

- 通过时间id的逆向工程进行时间序列的交叉验证，交叉验证用于防止模型过于复杂而引起的过拟合
- 采用KNN生成最邻近聚合特征
- 融合LightGBM, MLP和MoA' s 1D-CNN三种模型

Optiver 2nd place pipeline



均方百分比误差RMSPE
= 0.17573

TOP2 方案Pipeline

Ref: <https://www.kaggle.com/c/optiver-realized-volatility-prediction/discussion/274970>



Top2 方案介绍

特征工程

• 时序的逆向工程

逆向特征工程的含义是：恢复匿名数据中存在某些特定的关系

- 由比赛数据说明可知当前的竞价数据是标准化且匿名化的，每个在交易所交易的证券是以指定“价位”(最小价格变动单位)进行交易的，它代表价格可增减的最小幅度，并与该证券所处的价格区间有关，因此可以使用 tick size（价位：报价中的最小价格增量）恢复实际价格。
- 对time_id * stock_id的价格矩阵进行降维

$$Matrix = M^{T \times S}$$

- 上式为时间*股票个数矩阵，T为time_id的个数，S是股票的个数，矩阵中每个值是某个股票在某个时间点的 price。由于同一个time_id实际上有多个成交数据，所以是多维的。
- 使用t-SNE（一种非线性降维的机器学习方法，它能帮我们识别相关联的模式。t-SNE 主要的优势就是保持局部结构的能力。这意味着高维数据空间中距离相近的点投影到低维中仍然相近。）进行矩阵降维压缩到1维
- 总结来说，**tick是用于对原始数据进行点对点还原，t-SNE是找到价格之间的关联性还原序列。**



Top2 方案介绍

使用tick size进行训练集价格还原

```
def calc_price_from_tick(df):
    tick = sorted(np.diff(sorted(np.unique(df.values.flatten()))))[0]
    return 0.01 / tick

def calc_prices(r):
    df = pd.read_parquet(r.book_path,
                        columns=[
                            'time_id',
                            'ask_price1',
                            'ask_price2',
                            'bid_price1',
                            'bid_price2'
                        ])
    df = df.groupby('time_id') \
        .apply(calc_price_from_tick).to_frame('price').reset_index()
    df['stock_id'] = r.stock_id
    return df
```

```
def reconstruct_time_id_order():
    paths = glob.glob('/kaggle/input/optiver-realized-volatility-
prediction/book_train.parquet/**/*.parquet')
```

```
    df_files = pd.DataFrame(
        {'book_path': paths}) \
        .eval('stock_id = book_path.str.extract("stock_id=
(\\d+)").astype("int")',
             engine='python')
```

```
# build price matrix using tick-size
df_prices = pd.concat(
    Parallel(n_jobs=4)(
        delayed(calc_prices)(r) for _, r in df_files.iterrows()
    )
)
df_prices = df_prices.pivot('time_id', 'stock_id', 'price')
```

- 上述代码中，flatten函数的目的是将输入的df（同一个stock在同一个time_id下的价格）转为一维数组，然后对数组去重，去重后排序，排序后采用diff函数进行差值。由于diff函数的差值是默认的，所以这里获得的是time_id中从最后一个元素开始，后一位减前一位的差，并对差值进行排序并获得差值的最小值（sort函数默认升序），认为最小的时间差就是tick size价位。返回的是0.01/价位，即为股票的实际价格。
- 由上述的关键函数：求价位和重新赋值每个stock_id的价格时间序列，我们可以重新构建book_train订单簿数据的时序排序，将原始数据变为index = 'time_id' , column = 'stock_id' value = 'price' 的格式



Top2 方案介绍

t-SNE数据降维

- 使用非线性降维方式t-SNE将time_id*stock_id维的价格矩阵降到一维。通过t-SNE降维后能够保留数据之间的相关性，对于存在时间先后的时序信号有较大作用。

```
# t-SNE to recovering time-id order
clf = TSNE(
    n_components=1,
    perplexity=400,
    random_state=0,
    n_iter=2000
)
compressed = clf.fit_transform(
    pd.DataFrame(minmax_scale(df_prices.fillna(df_prices.mean()))))

order = np.argsort(compressed[:, 0])
ordered = df_prices.reindex(order).reset_index(drop=True)

# correct direction of time-id order using known stock (id61 = AMZN)
if ordered[61].iloc[0] > ordered[61].iloc[-1]:
    ordered = ordered.reindex(ordered.index[::-1])\
        .reset_index(drop=True)

return ordered[['time_id']]
```

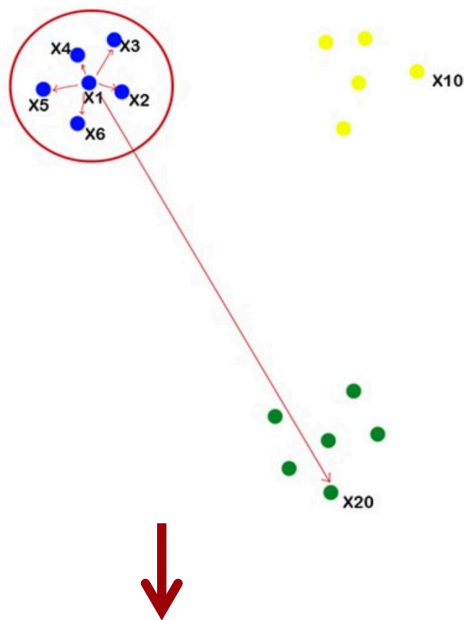
- 对于输入的价格矩阵，用价格的均值填充空值，之后对矩阵元素进行归一化（minmax函数），采用t-sne算法降维。
- n-component=嵌入空间的维度，这里表示将其降到1维。perplexity是控制数据点是否适合算法的主要参数，推荐范围是（5-50）。这里选择400，相对较高，说明算法更偏向于考虑全局结构。
- 降维后得到的数组用argsort函数进行从小到大的排序并输出索引，降维后得到的是应当还原的时间序列。



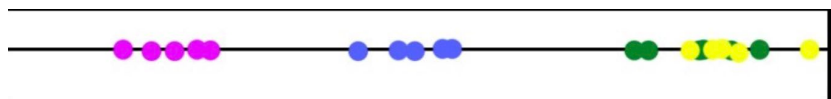
Top2 方案介绍

t-SNE原理

- 首先，它将通过选择一个随机数据点并计算与其他数据点 ($|x_i - x_j|$) 的欧几里得距离来创建概率分布。从所选数据点附近的数据点将获得更多的相似度值，而距离与所选数据点较远的数据点将获得较少的相似度值。使用相似度值，它将为每个数据点创建相似度矩阵 (s_1)。其次，它将根据正态分布将计算出的相似距离转换为联合概率。



- 由上图可知，我们可以说 x_1 的邻域 $N(x_1) = \{x_2, x_3, x_4, x_5, x_6\}$ ，这意味着 x_2, x_3, x_4, x_5 和 x_6 是 x_1 的邻居。它将在相似度矩阵“ s_1 ”中获得更高的价值。这是通过计算与其他数据点的欧几里得距离来计算的。 x_{20} 远离 x_1 。这样它将在 s_1 中获得较低的值。
- 通过以上的计算，t-SNE将所有数据点随机排列在所需的较低维度上。t-SNE将再次对高维数据点和随机排列的低维数据点进行所有相同的计算。但是在这一步中，对于高位数据根据正态分布分类概率，对于低维数据根据t分布分配概率，减少拥挤问题。

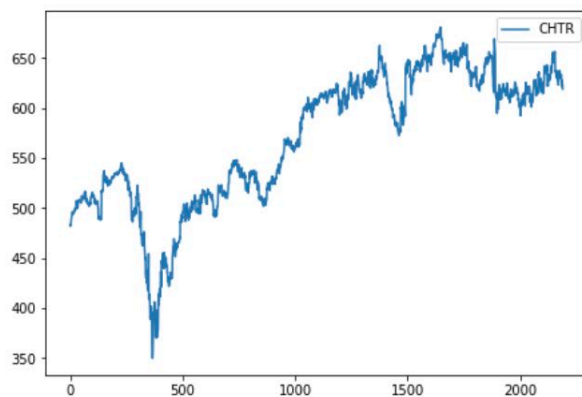
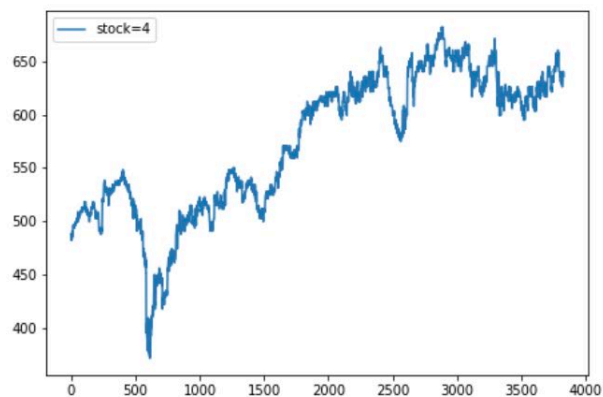




Top2 方案介绍

时间序列还原结果

- 通过与真实市场数据进行比较，可以很容易地验证训练数据的时序恢复的正确性。例如对于stock_id = 4的股票，可以通过对比真实市场数据发现，训练数据是2020/1/1到2021/3/31之间的数据。左：t-SNE恢复的股价右：yFinance检索的实际股价(2020-01-01~2021-03-31)



数据集生成方式猜测

- 题目给出的数据是2020.1.1到2021/3/31，在这期间开市了443天，题目给出了在这期间的3830个时间ID，这表示每天会被记录 $3830/443=8.6$ 个时间ID。假设optiver排除了股市开盘第一个小时和闭市前最后一个小时，那么每个交易日的10:30-15:00用于生成竞价数据，如果每30分钟记录数据，那么每天最多记录9个时间ID，考虑到提前关闭等特殊情况，因此符合每天8.6个时间ID的记录。



Top2 方案介绍

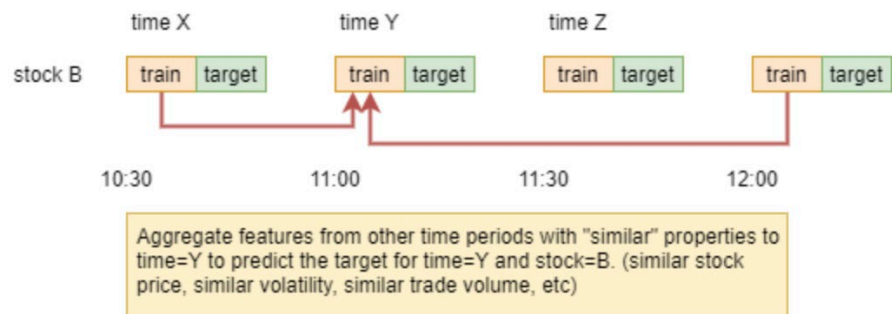
特征聚类

本方案中特征工程的核心在于相似性聚类：

- **时间相近：**如果我们考虑数据是如何生成的，那么在一个time-id的目标计算周期之后仅仅10分钟，就会开始下一个time-id的训练数据。相近时间的RV会对需要预测的target RV具有一定的指示作用
- **环境相似：**在市场具有相似价格，波动率和成交量时，同一个股票的RV对于需要预测的特定时间的RV具有一定的指示作用

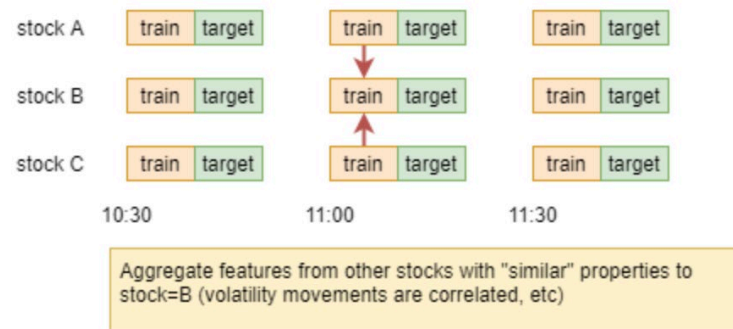
因此使用不同的距离度量方式选择最近邻的N个time_id (N=2,3,5,10,20,40)，并计算这些点特征的平均值，如RV和成交量等。除了在横向上对时间维度进行聚合，该方案对纵向上相似的stock_id也进行了聚合，例如20个具有相似波动率的股票在5个最邻近的time_id上平均价格。聚类方法使用了KNN（最邻近搜索的无监督分类方式）。

time-id nearest neighbor



以stock = B time = Y为例，聚合具有与time=Y“相似”属性的其他时间段的特征，(相似的股价、相似的波动性、相似的交易量等)

stock-id nearest neighbor



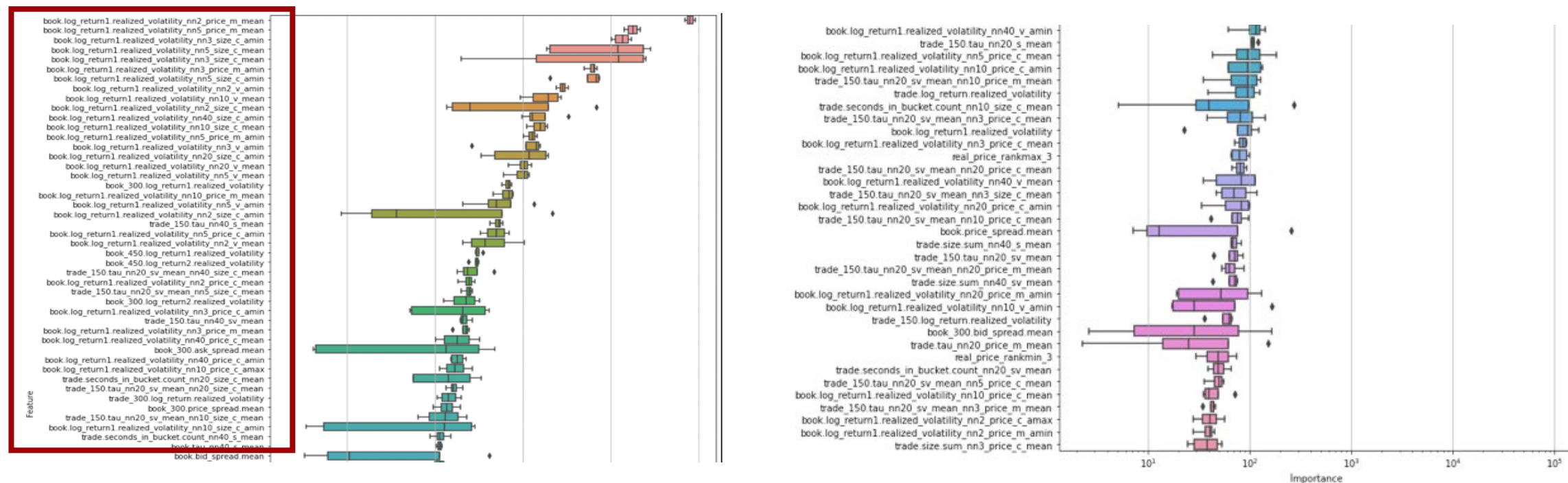
聚合其他股票在相近时间端的相似特征，（波动性移动等特征）



Top2 方案介绍

特征重要程度可视化

方案总共创建了584个特征，其中362个是最近邻特征，根据作者Notebook，得分提高大部分都是基于这些NN特征，而且从真实场景来看，最近邻聚合附近特征的想法可能可以在真实的模型中使用。下图所示为特征的重要程度，基于订单簿数据的最近邻特征对于模型的贡献程度较大。



重要程度排名较高的均为nn特征



Top2 方案介绍

构造验证集

- 测试集中并没有使用时间逆序的方式，因为不能够保证t-SNE能够完全还原数据，从而在测试集引入一些不必要的误差。但是可以基于时间还原后的训练集建立验证集，解决可能存在的过拟合问题，提升泛化能力。
- 时间序列交叉验证(CV)。已知时间戳的正确顺序可以构造验证集
- 协变量移位的检测。使用对抗验证（adversarial validation）等方法找到真实的随时间变化的特征。对抗验证会使得训练集和测试集的分布是一致的，来解决训练集上效果很好，但测试集效果很差的问题。

时间序列交叉验证

- 两种最常见的交叉验证方式分别是 k 折交叉验证和 hold-out 交叉验证。但时序问题中不能用k折交叉验证，因为数据是有顺序依赖的。
- 时间序列模型的交叉验证可以采用的方法是滚动式的交叉验证。从一小部分数据子集开始进行训练，对后面的数据点进行预测，然后检查预测数据点的准确性。然后将相同的预测数据点作为下一个训练数据集的一部分，对后续数据点进行预测。

D1	D2	D3	D4	D5	D6	D7	D8	D9
D1	D2	D3	D4	D5	D6	D7	D8	D9
D1	D2	D3	D4	D5	D6	D7	D8	D9
D1	D2	D3	D4	D5	D6	D7	D8	D9
D1	D2	D3	D4	D5	D6	D7	D8	D9

- Training: [1] Test: [2]
- Training: [1, 2] Test: [3]
- Training: [1, 2, 3] Test: [4]
- Training: [1, 2, 3, 4] Test: [5]

- 在时间序列交叉验证中，每天都是测试数据，前一天的数据是训练集。D1，D2，D3等是每天的数据，蓝色突出显示的日期用于训练，黄色突出显示的日期用于测试。在本模型中，采用了4折的时间序列交叉验证，每个fold都有10%的数据进行验证



Top2 方案介绍

对抗验证

- 对抗验证（Adversarial Validation），并不是一种评估模型效果的方法，而是一种用来确认训练集和测试集的分布是否变化的方法。它的本质是构造一个分类模型，来预测样本是训练集或测试集的概率。从而使得本地的测试数据和最终在私榜上的排名更加稳定。
 - 构建一个样本的分类器，该二分类器的任务用于区分样本来源于训练集，还是测试集。
 - 将新的训练数据集进行划分，保留部分样本作为该样本分类任务的测试集，利用分类算法(XGBoost, LightGBM)等对数据集进行训练，AUC作为模型指标；
 - 在测试集中进行验证，如果模型效果AUC在0.5左右，说明原始数据中训练集，测试集分布是一致的；如果AUC较大，如0.9说明样本分类器很容易区分样本，训练集与测试集存在很大差异；
 - 根据第3步的结论，对于分布一致的，正常对目标任务训练即可。对于分布不一致的，可以继续进行样本挑选的尝试。利用上述样本分类器模型，对原始的训练集进行打分预测，并将样本按照模型分从大到小排序，模型分越大，说明与测试集越接近，那么取训练集中的TOP N 的样本作为目标任务的验证集
- 通过**基于时间序列的对抗验证**，发现非常多的特征随着时间的变化影响很大，例如order_count和total_volume这些，所以方案中**将其转化为在某个时间点的rank（处理为全局相对值）进行处理**



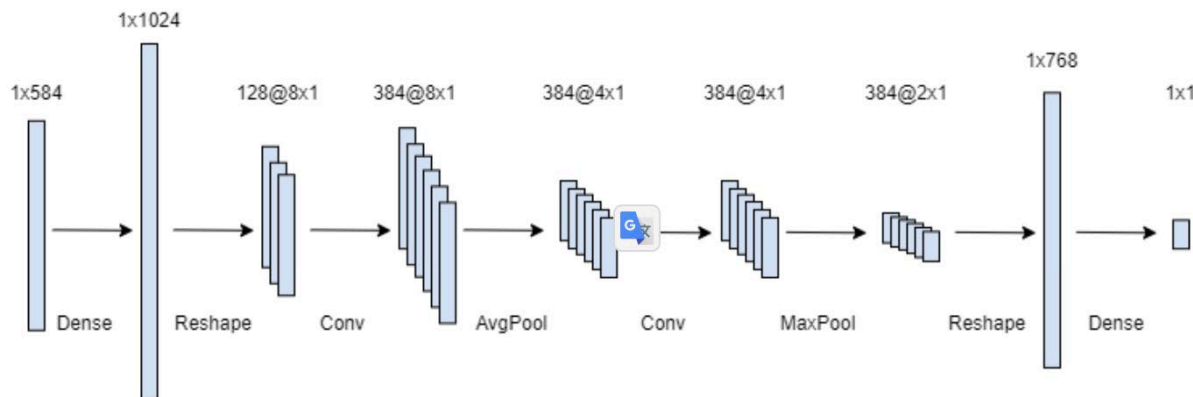
Top2 方案介绍

预测模型

使用了三种简单的模型进行预测：LightGBM、1D-CNN和MLP。该方案中没有使用预训练的模型，直接使用测试数据进行特征计算，每个模型都用10个不同的seed训练，选出了预测验证分数最高的5个模型。在模型融合方面主要进行了加权平均的方式，LightGBM：MLP：IDCNN=4：1：4

比较新颖的是1DCNN，这是Kaggle中一个药物作用机理预测比赛2nd的解决方案，**核心思路是将表格数据变为图像数据的形式再采用CNN的方式进行预测。**

- 先通过全链接层增加特征维数M。该层的作用包括通过增加尺寸为图像提供足够的像素，并通过特征排序使生成的图像有意义。接下来，数据被直接reshape为图像格式（尺寸 $N \times 1$ ，channel数为C），注意： $C \times N = M$ 。之后采用多个一维CNN抽取特征。最后，利用提取的特征展开并且使用FC层进行拼接预测。



1D-CNN architecture

- 基本数据特征217个
- 纵向stock_id聚类特征 60个
- 横向time_id聚类特征 302个
- 其他特征 Misc Feature 5个

Ref about ID-CNN: https://mp.weixin.qq.com/s/_K5OA3qyZYgFLM1NG97MWA



Top2 方案介绍

结论：没有用的部分

测试过程中没有用的模型：

- 一些特定领域的特征，如用来衡量个别股票或股票基金相对于整个股市的价格波动情况的 β 系数
- TabNet效果可以，但是训练时间过长
- 训练关于残差的神经网络
- 降维的特征（只针对一条数据建立的简单特征）

结论：可以继续完善的部分

还没来得及做，但是感觉有用的工作：

- 300秒模型(将训练数据分成前半部分和后半部分，并创建一个模型，从前半部分预测后半部分的RV，并将其用作元特征（meta-feature）或数据增强)
- 结合LSTM和RNN，这些都是常用的用于时序的模型
- 通过AutoEncoder进行表示学习用于降维，尽可能多地将信息编码到特征中



北京大学量化交易协会