

大学生助学金精准资助预测初赛答辩

Yancy&Zhendong 团队

团队成员

杨轩

新加坡国立大学 硕士

刘振东

新加坡国立大学 硕士

曹峻许

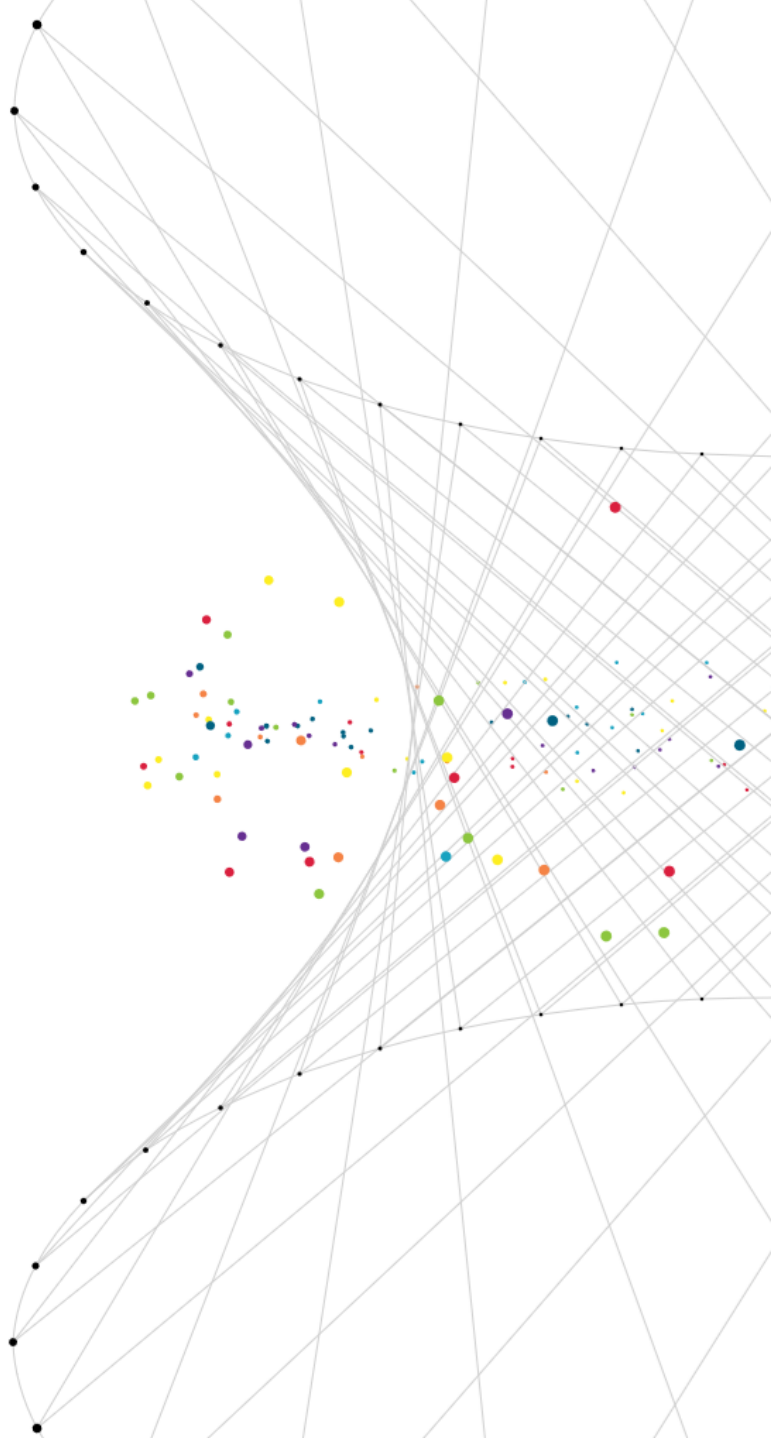
华南理工大学硕士

刘一鸣

卡内基梅隆大学 硕士

卢健

香港科技大学 硕士



内容提要

CONTENTS

流程分析

PART ONE

特征提取

PART TWO

模型&调参


PART THREE

模型融合

PART FOUR

感想和结论

PART FIVE



流程分析

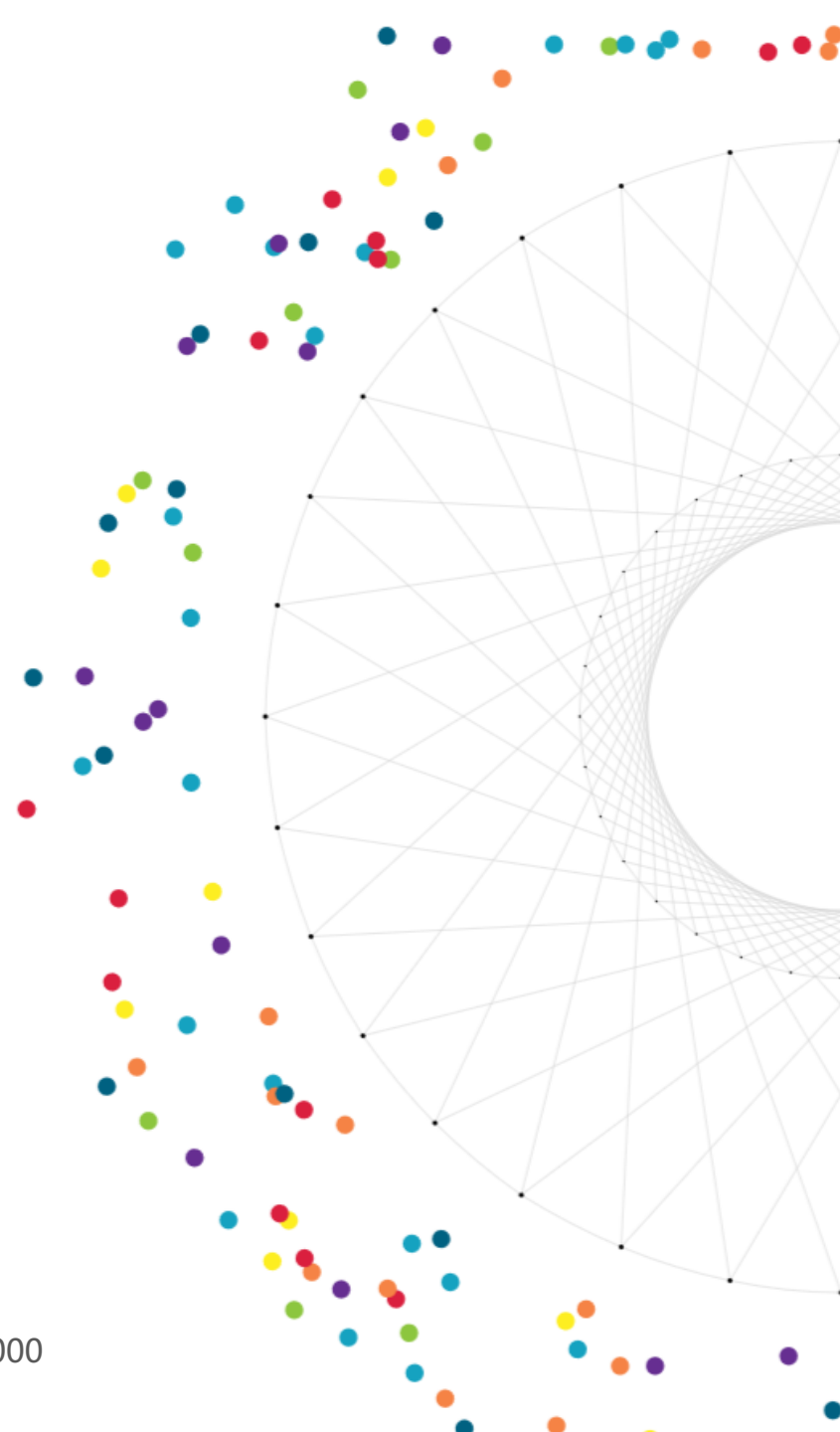
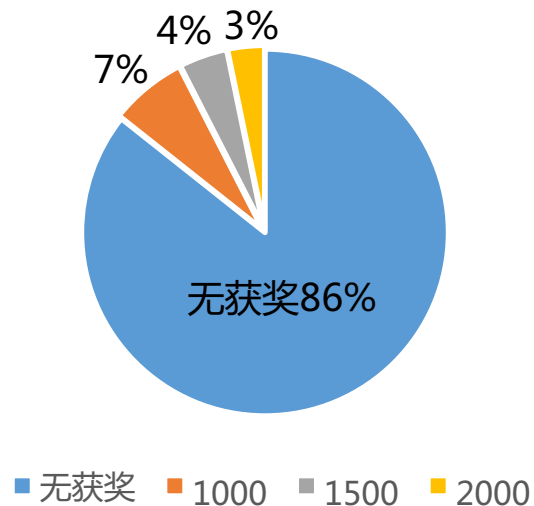
PART ONE

PART ONE 流程分析

本次比赛以《大学生助学金精准资助预测》为题，提供学生的一卡通等日常行为数据，要求参赛者预测每个学生的获奖情况。我们将这个问题看待成一个典型的多类别数据不平衡分类的问题去解决。

- 数据集---学生日常行为信息
 - 校园卡消费记录 Card.txt
 - 宿舍门禁记录 Dorm.txt
 - 图书馆门禁记录 Library.txt
 - 图书借阅记录 Borrow.txt
 - 成绩信息 Score.txt
- 多分类
 - Label : {0, 1000, 1500, 2000}
 - 各类别人数比例不平衡

训练集获奖分布



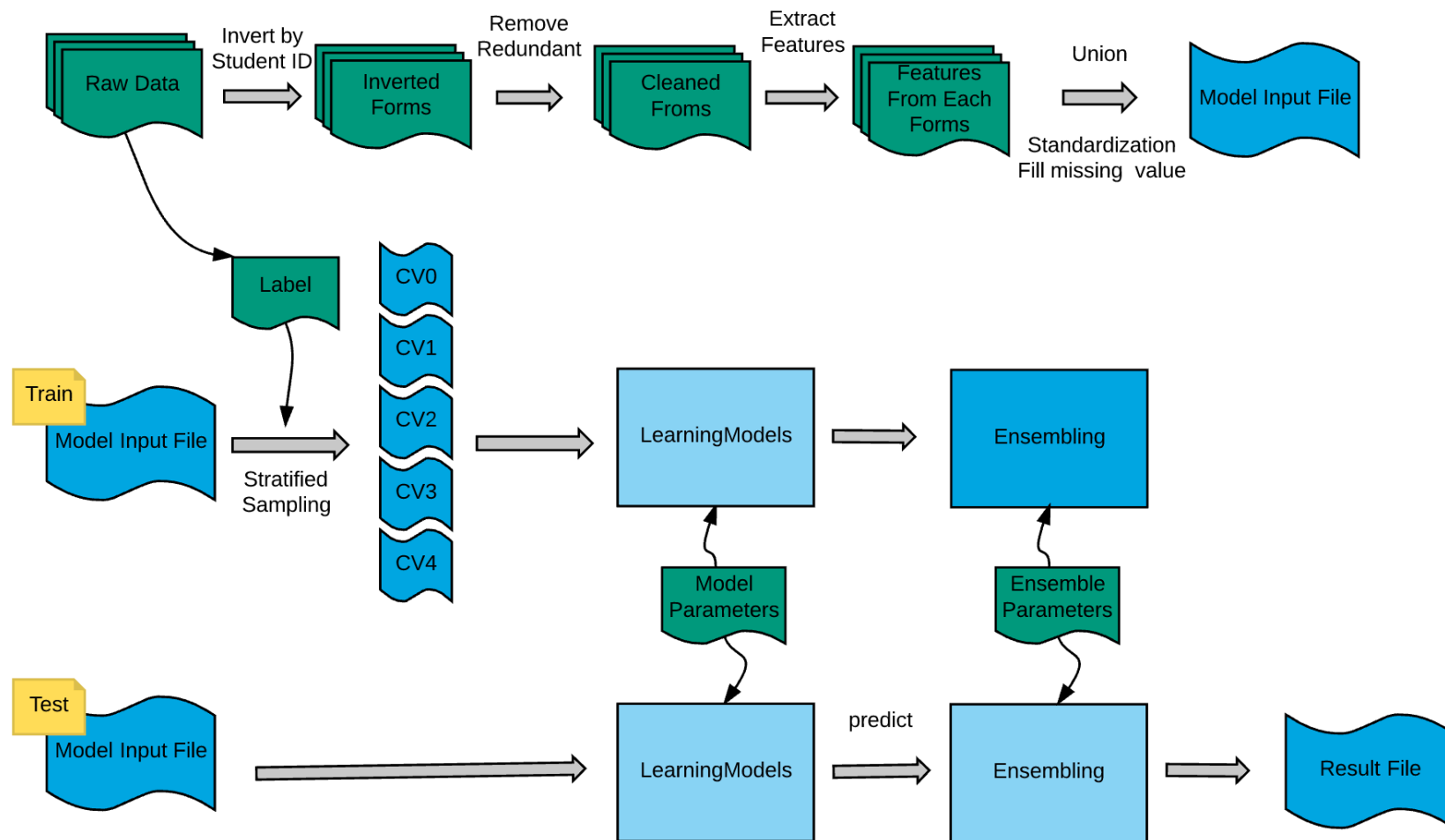
PART ONE 流程分析

主体结构示意图 Architecture of The System

处理数据

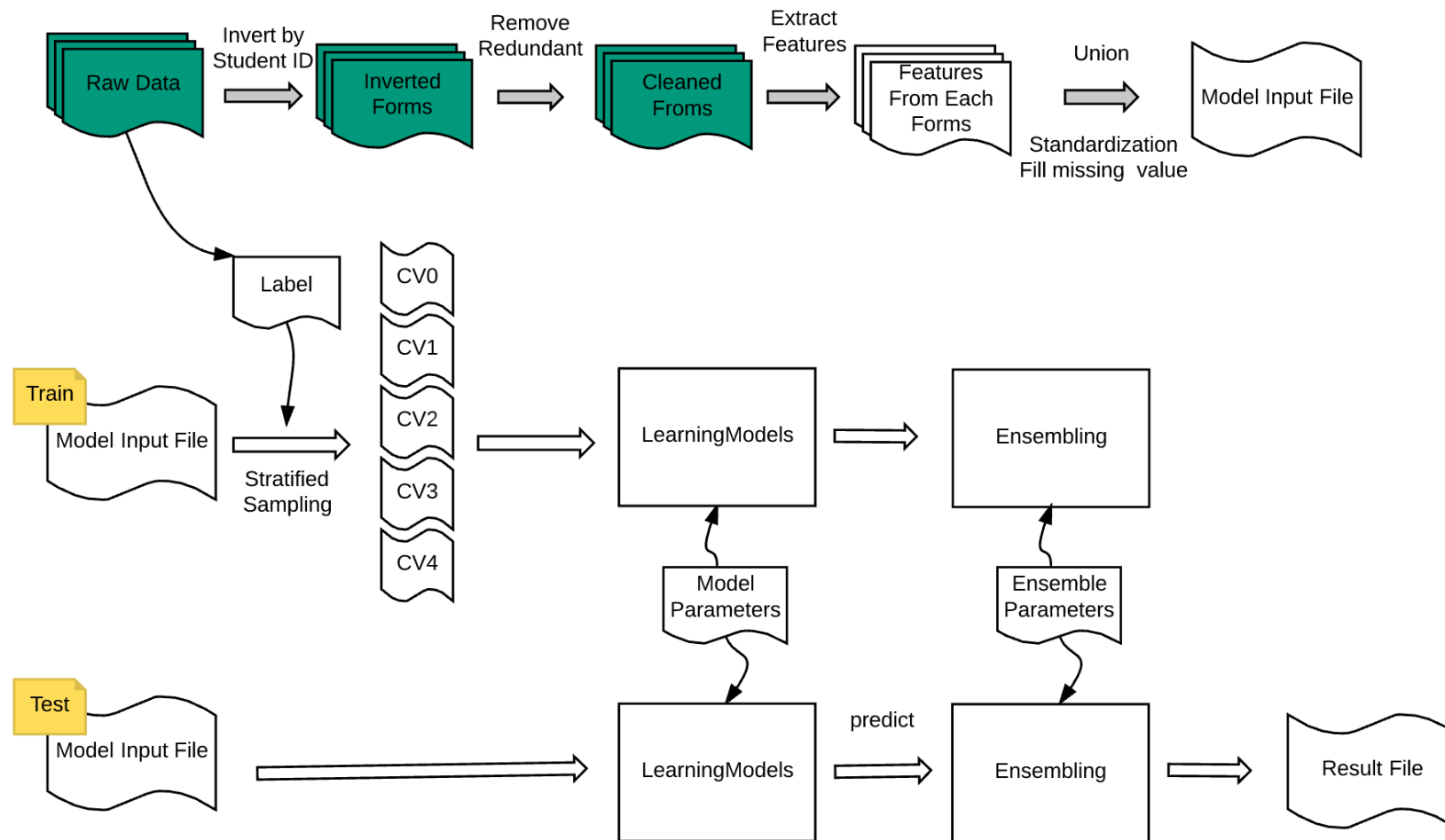
训练模型

预测结果



格式转换与数据清洗


Invert and Clean the Data



格式转换与数据清洗

Invert and Clean the Data

- 数据格式转换
 - 将原始表转换为key - value的格式，key为学生id，value为该学生的所有行为记录，方便以后快速的进行数据清洗和特征抽取。
- 数据清洗
 - 去除掉消费表中的重复条目
 - 处理借阅表不规整条目

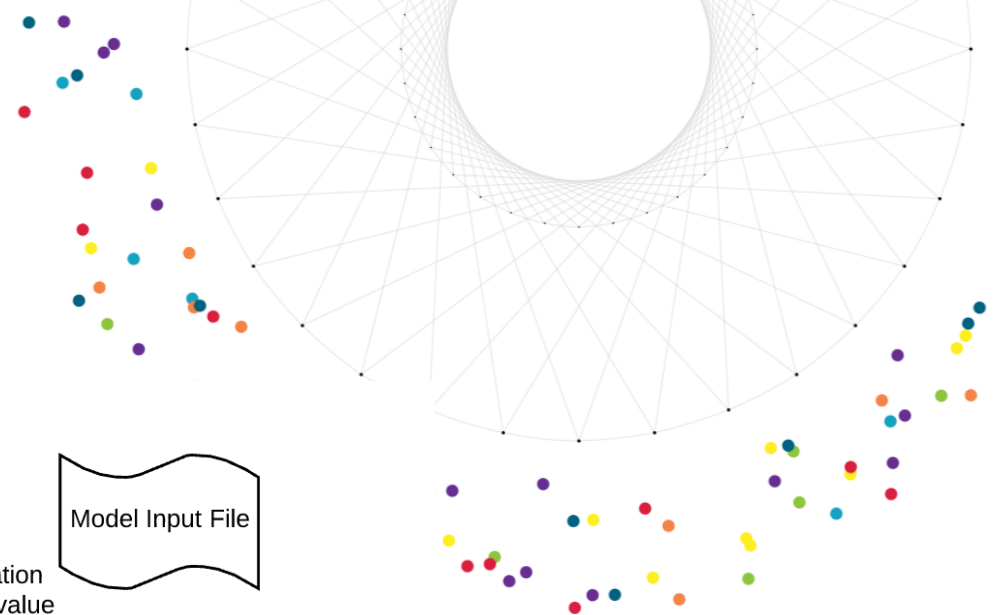
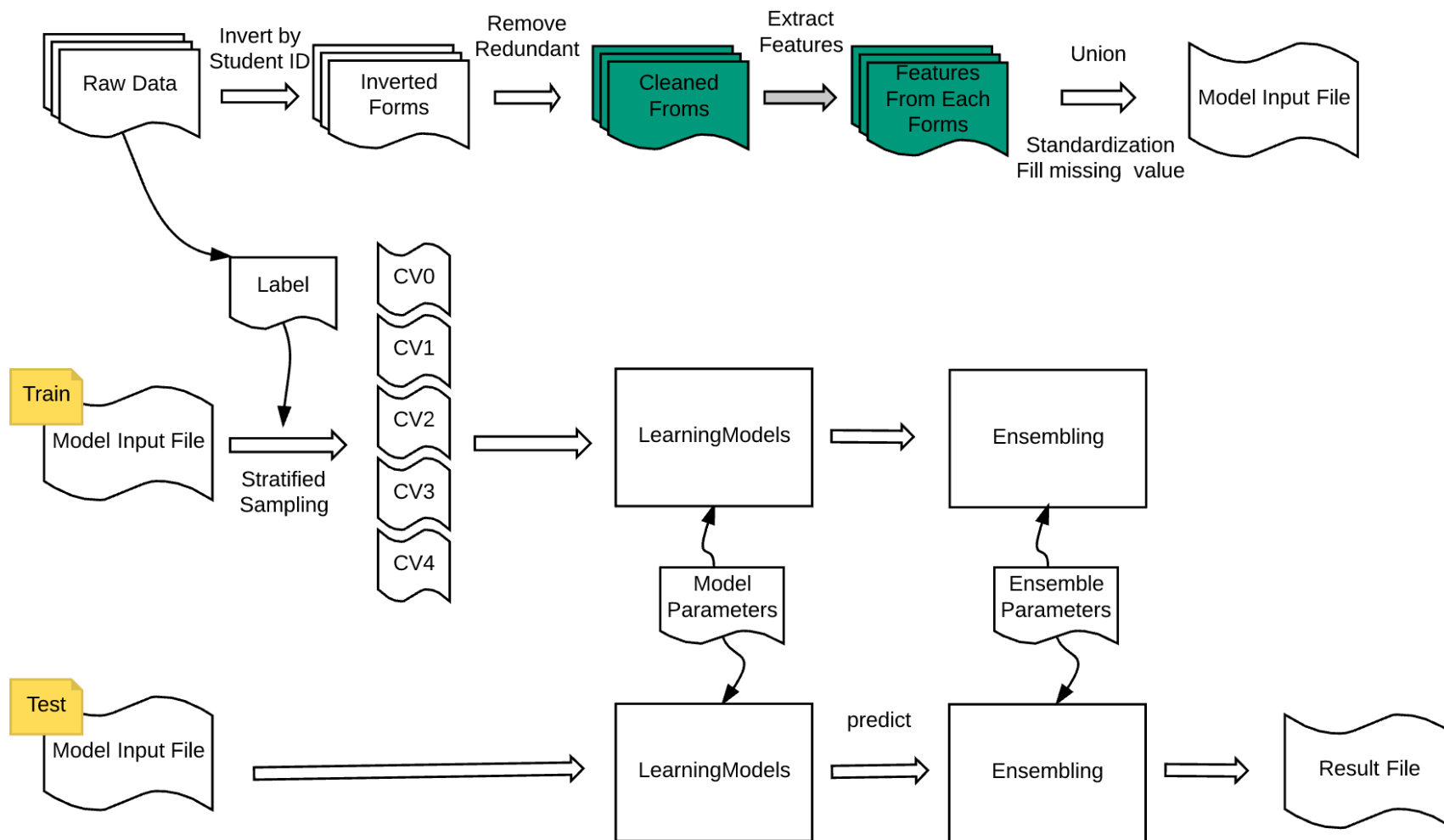


特征提取

PART TWO

特征提取

Feature Extraction



特征提取

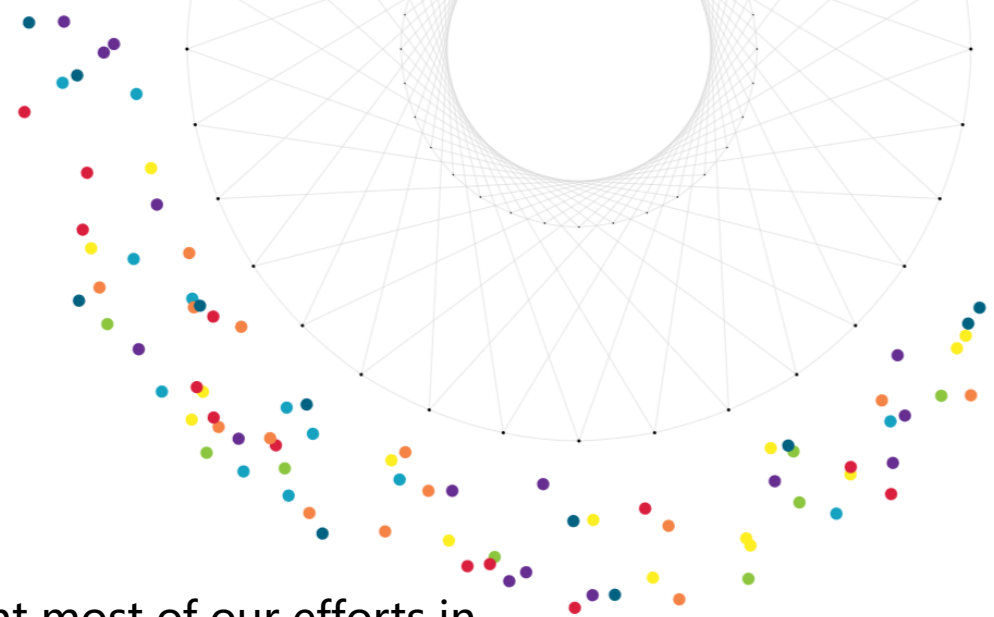
Feature Extraction

“The algorithms we used are very standard for Kagglers. We spent most of our efforts in feature engineering. We were also very careful to discard features likely to expose us to the risk of over-fitting our model.” 对于Kaggle的参赛者们来说，所使用的算法已经相当固定了，我们把更多的精力花在了特征工程上。我们很珍惜每一个feature，所以即使当我们删掉一些过拟合feature的时候我们也会非常小心。

— Xavier Conort （ Kaggle 个人排行榜历史第一名，18金6银3铜 ）

“...some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.”有些机器学习项目成功了，有些失败了，是什么造成了这种差异？最常见最有可能的重要因素就是它们所使用的特征。

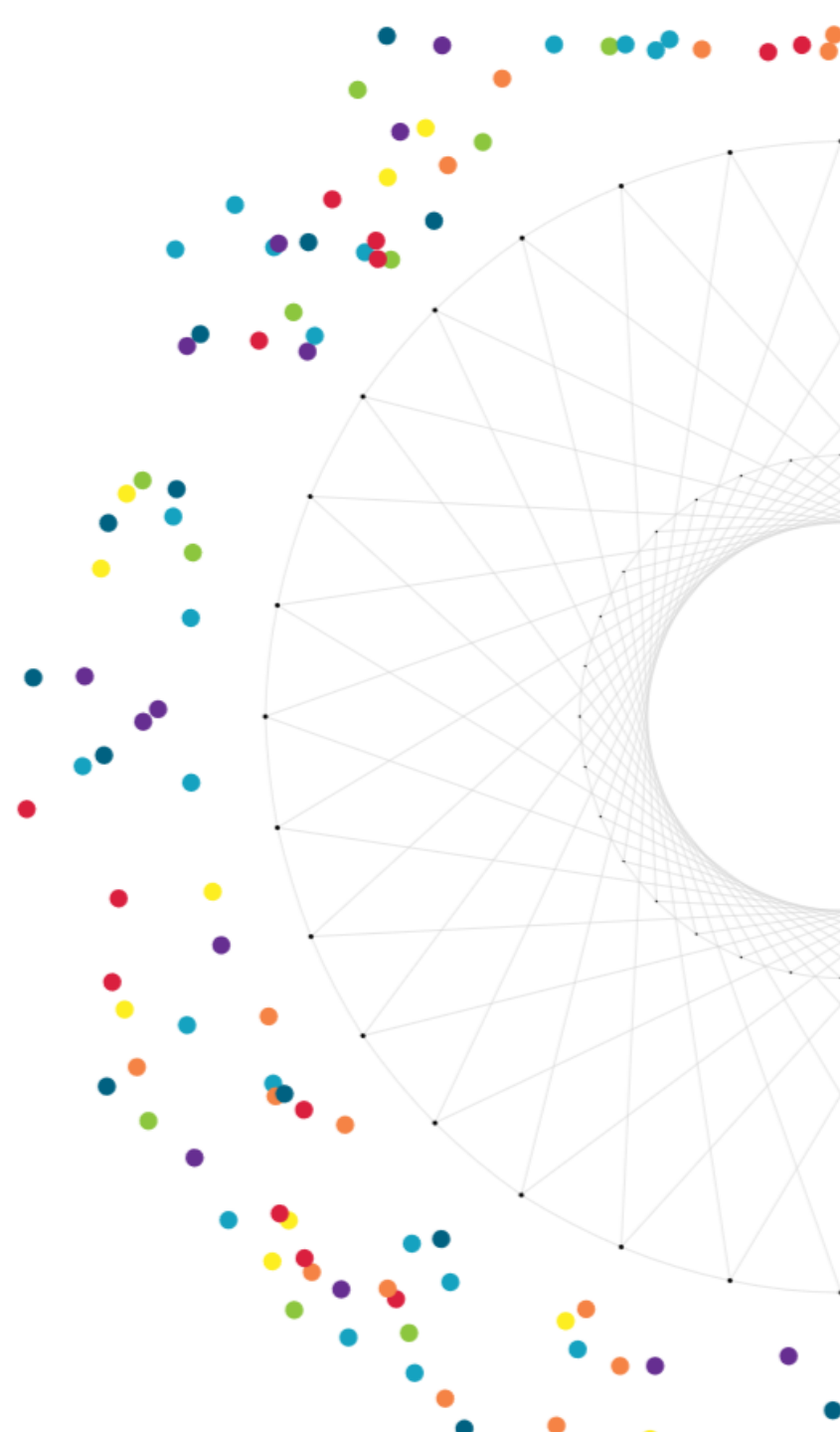
— Pedro Domingos （ 美国华盛顿大学计算机系教授， Markov logic network的作者 ）



我们是如何做的？

How did we do it?

- 地毯式特征抽取
 - 对每个表，每个列都要进行特征抽取
- 尽可能在设计特征的阶段保证合理性
 - 多人论证，设计feature
 - 查看原始数据，验证想法
- 迭代的抽取
 - 第一轮：200个feature，线上最高排名128
 - 第二轮：500个feature，线上最高排名14
 - 第三轮：1151个feature，线上排名保持在前3
 - 第四轮：1200个feature（舍弃）



特征展示

Present Our Features

图书借阅

- 是否借书
- 借阅书籍数量
- 借阅考研、编程类、托福gre雅思不同种类书籍的借阅次数

图书馆门禁

- 不同时间段进出图书馆的次数
- 晚上进出图书馆的次数
- 进出总次数
- 周末进出图书馆的次数
- 去图书馆天数

宿舍门禁

- 不同时间段进出宿舍次数
- 每天最早、最晚离开宿舍平均时间
- 平均每月在宿舍的最大天数
- 每天进出宿舍的次数
- 周末进出的次数

成绩排名

- 学生成绩排名
- 成绩排名百分比
- 学院各个获奖类别人数比例



第二轮

第三轮

校园卡消费

其他

过完年后第一笔消费的日期（反映返校早晚）
每天总消费在0-10元、10-20元、等区间的次数

地点维度

前十大最受欢迎的地点

用户去的最多的前十大地点

用户花钱最多的前十大地点

单价最高的前10，50，100，200，300个地点

...

时间维度

24个小时

暑假，节假日，周末

早中午餐时间

用户卡充值前后10天

...

消费方式维度

12种消费方式（食堂、超市、开水等）

消费种类（卡充值，换卡，支付领取等）

...

每笔

每天

统计量

计数

最大 / 小值

总额

方差

均值

计数

每天第一条记录和最后一条记录的平均时间，最早时间，最晚时间

时间间隔

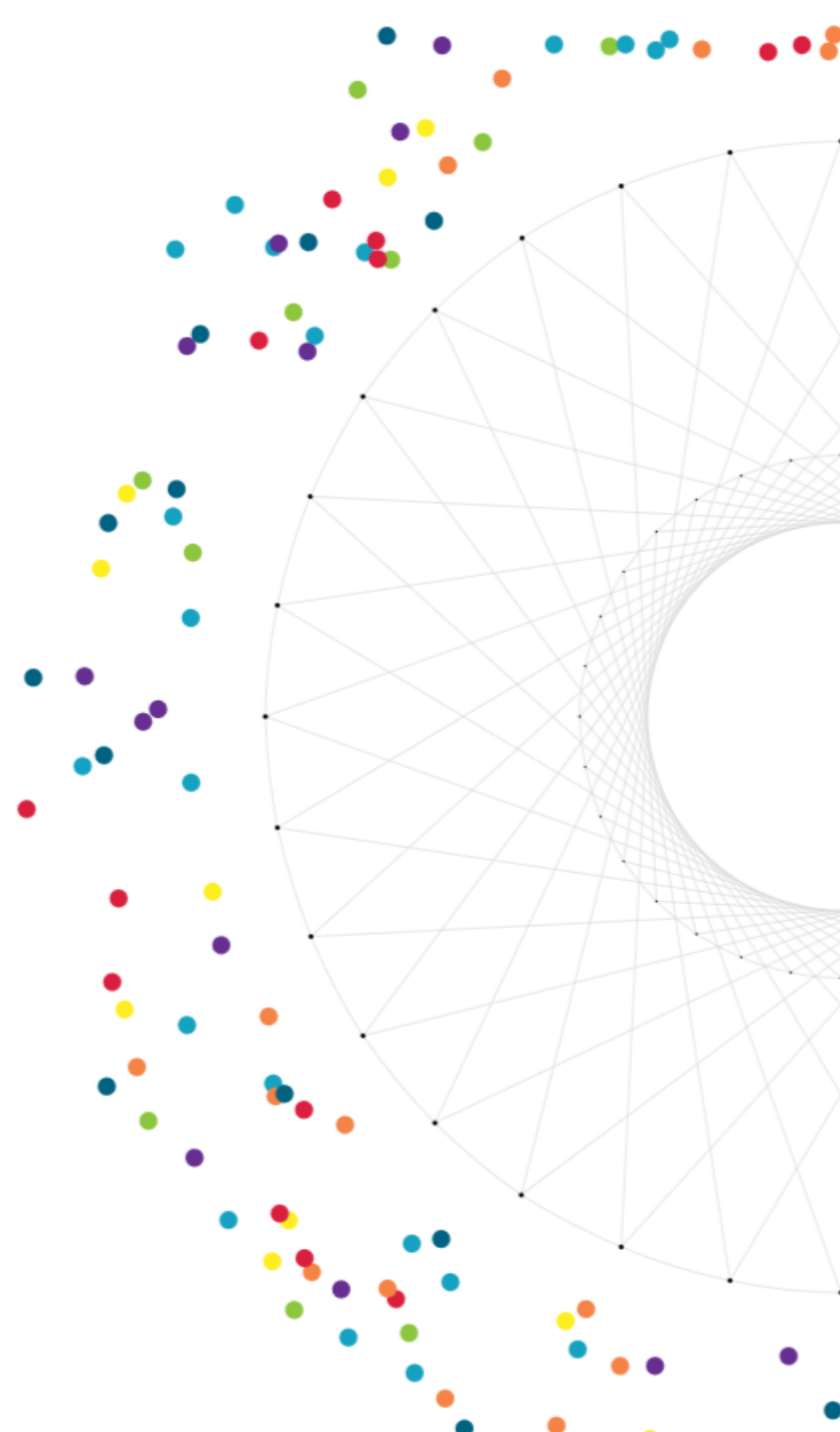
中位数

涉及天数

涉及的地点数量

特征构造 Feature Construction

- 消费总额或者计数等指标除以活跃天数
 - 活跃天数指的是用户有记录的天数，不同用户活跃天数差别很大，所以除以活跃天数会使得特征更加公平
- 用户在本学院消费排名（倒序）乘以成绩排名（正序）
- 用户不同种类的消费额除以消费总额（比如食堂消费除以总消费，反映用户不同消费种类的占比）
- 周末消费的平均值减非周末消费的平均值
- “圈存转账” “卡充值” “支付领取” 三种充值类特征相加
- ...



特征重要性排名一览 Feature Importance Ranking

Random Forest给出的前20个重要的特征

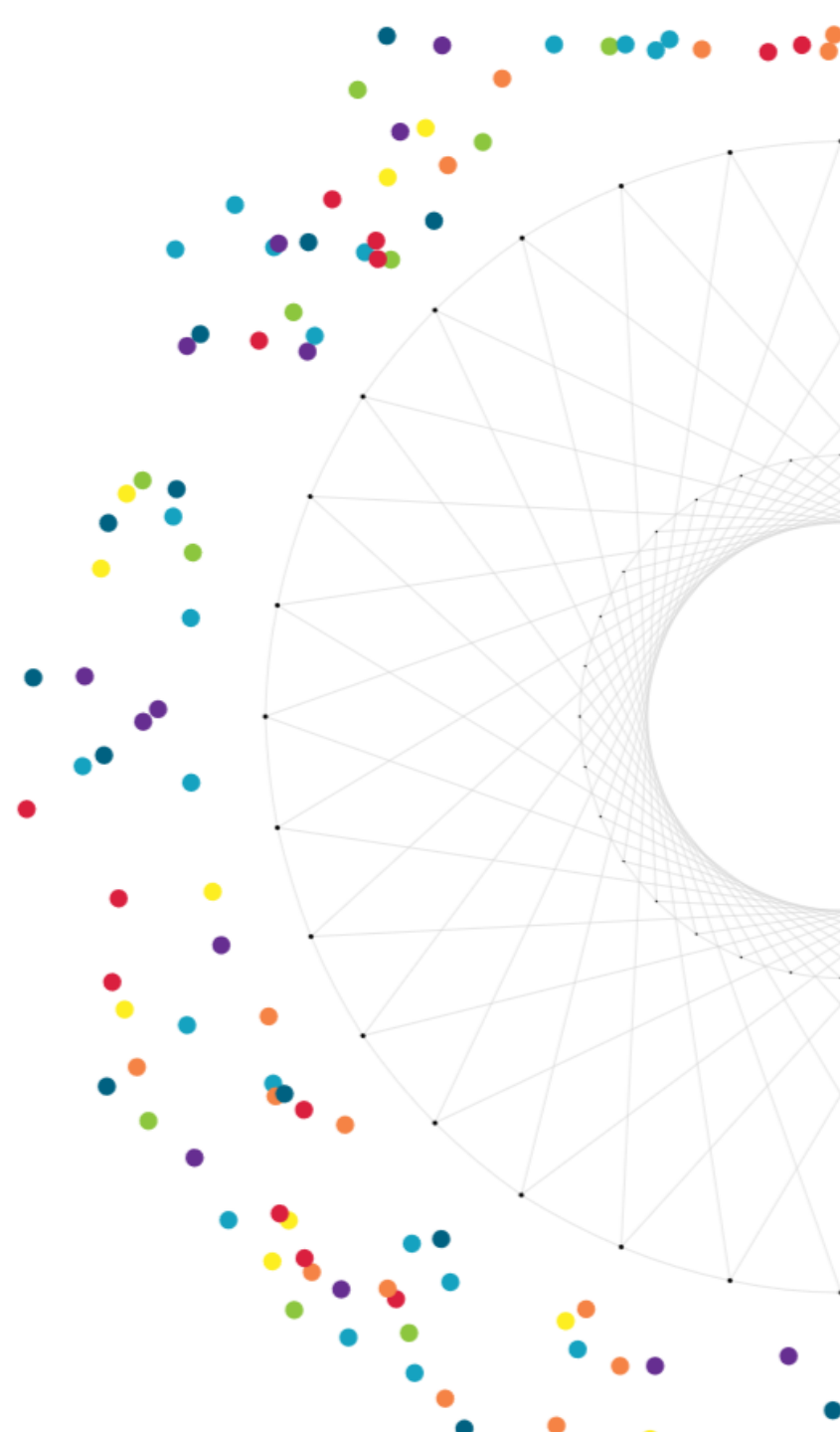
1	是否曾更换校园卡	if_change_card
2	每天总消费在0-10元范围的次数/活跃天数	consumeDayOver_0
3	每日7点-8点消费的总额/活跃天数	hour_07_sum_per_day
4	成绩排名乘以消费排名的值	rank_score_consume
5	每日6点-7点消费的总额/活跃天数	hour_06_sum_per_day
6	该学生所在学院获2000助学金的人数比例	2000_percent
7	该学生所在学院获1000助学金的人数比例	1000_percent
8	金额在0-2.5元之间的消费笔数/活跃天数	slice_0_2.5
9	学生的成绩排名/学院人数	rankPercent
10	卡充值总额/活跃天数	kachongzhi_top_up_sum_relative
11	每日消费总额的方差	all_consume_one_day_var
12	该学生消费的相对总额在用户所在学院的排名	rank_in_faculty
13	该学生每天吃早餐的平均时间	breakfastTime_ave
14	每日6点-7点消费的总额	hour_06_sum
15	学生成绩排名的值	absoluteRank
16	每日7点-8点消费额的最大值	hour_07_max
17	该学生所在学院获1500助学金的人数比例	1500_percent
18	在饭堂每日消费额的方差	canteen_consume_one_day_var
19	该学生消费金额排名第2高的地点消费的次数	top_amount_place_2_count
20	每天11点-12点消费的次数	hour_11_count

在树形模型中，一个feature被用作分裂结点的次数越多，则此feature越重要。

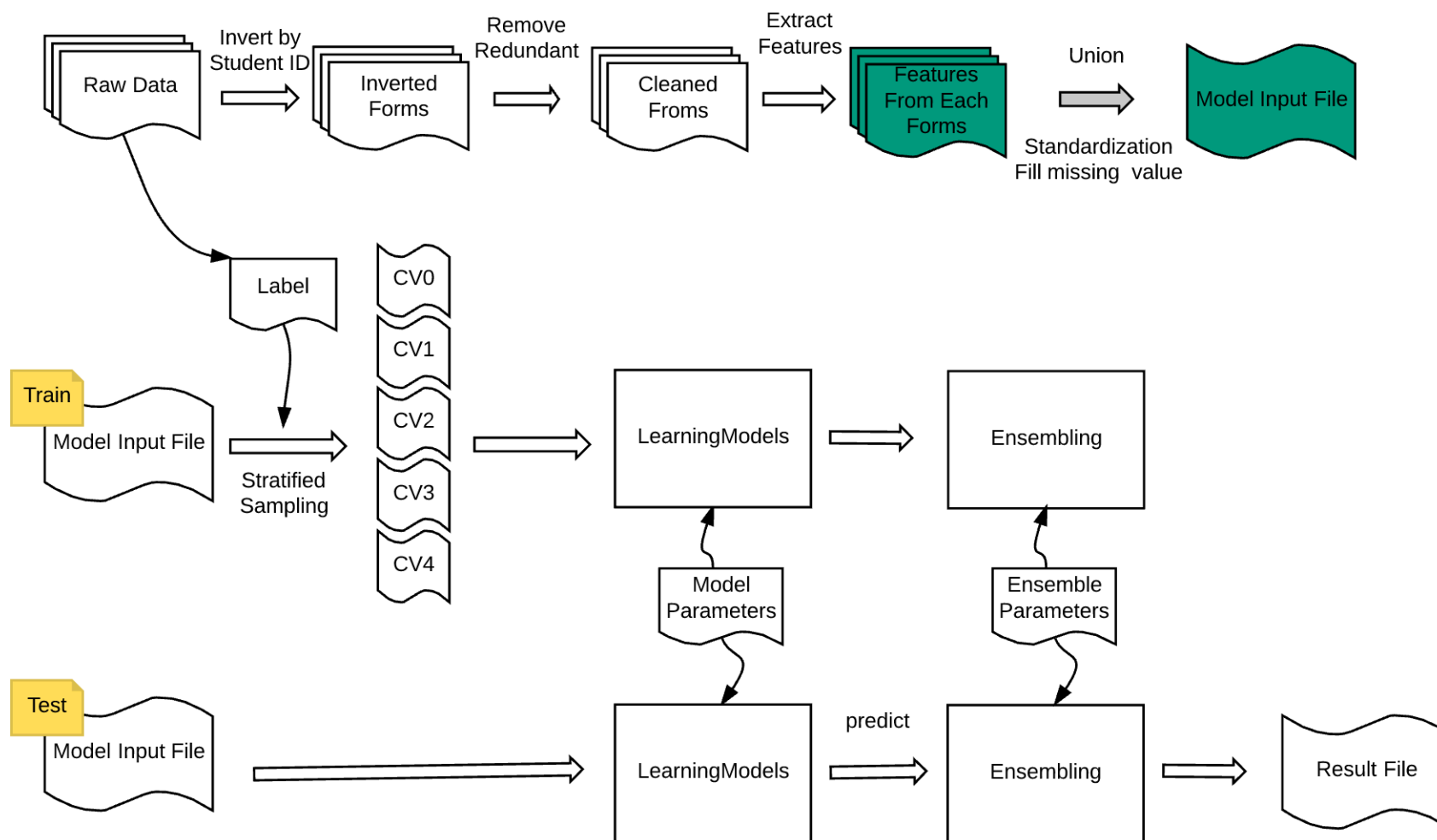
特征选择

Feature Selection

- 经过实验，只有排名靠前的特征对模型影响的结果较大
- 删除准则：重要性靠前的，但是人眼观察怀疑为导致过拟合的特征。
- 验证方法：删除特征之后，在验证集上重新训练模型，观察分数是否上升。
- 我们所删特征：用户借阅不同类别书籍的特征，用户所在学院特征，周一至周五每天消费的统计量特征



预处理 Pre-processing



预处理 Pre-processing

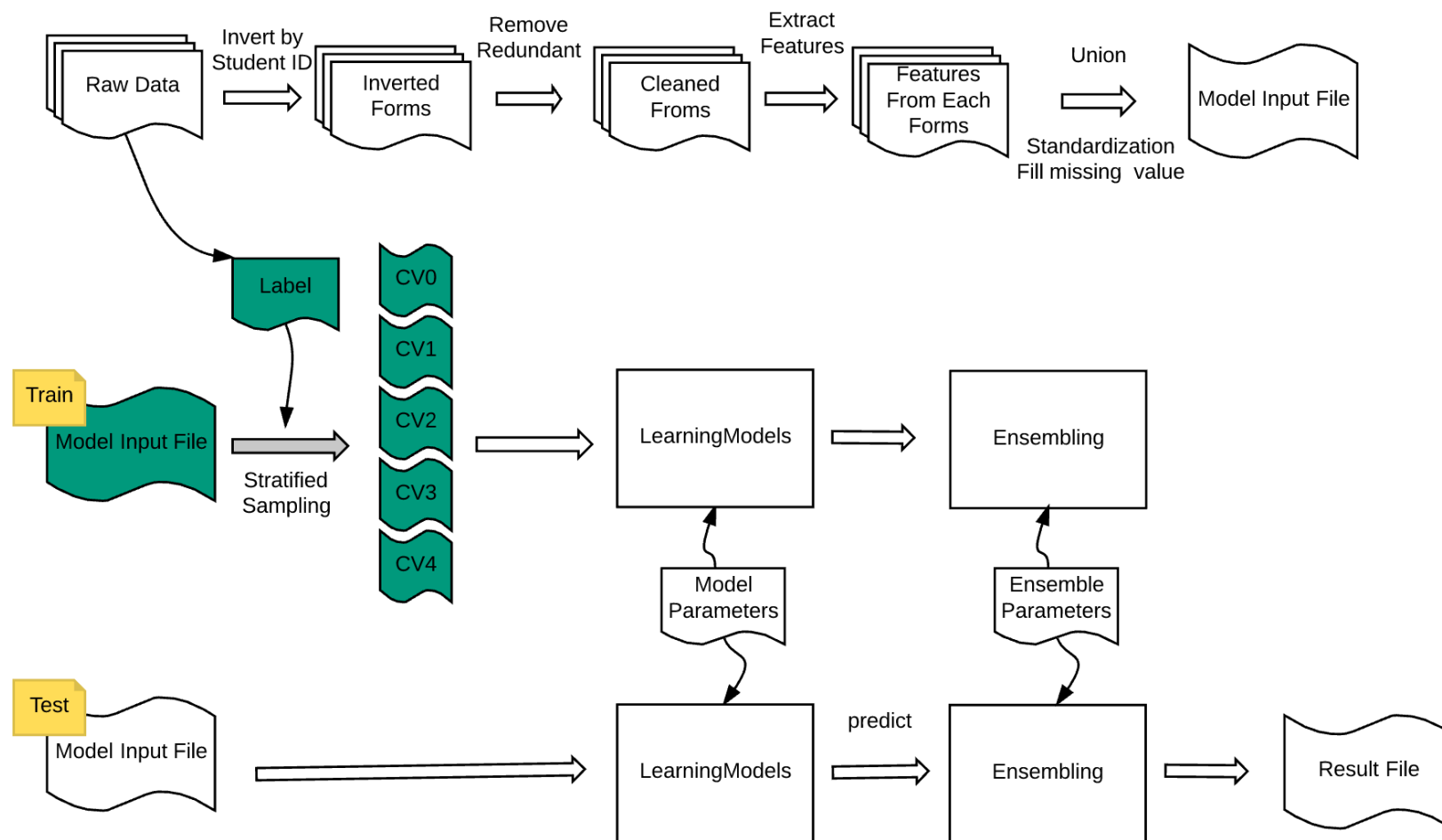
- 缺失值填充
 - 根据每个特征的不同性质使用不同的值进行填充
 - 如图书借阅数量这个特征，我们将其填充为0
 - 如学生1:00-2:00时间段消费总额这个特征，若用户在原始表中没有任何消费记录，则认为这是系统丢失用户信息的情况，填充为-1；如果用户在其他时间段有消费记录，我们就填充为0
- 标准化
 - $$Z = \frac{x - \mu}{\sigma}$$
 - 所有值减去平均值，除以标准差
 - 标准化完全不会影响基于树（Random Forest, Ada Boost等）的模型效果和效率，但是会提升SVM、Neural Network等算法的效率和准确度

预处理 Pre-processing

- 欠采样
 - 从无获奖用户中进行随机采样，缺点是丢失数据。
- 过采样
 - 复制稀有样本，使各种类数量达到一致，缺点是增加了样本数量会增加模型训练的时间，而且只能成倍的增加样本的权重
- 设置样本权重
 - Minimize $\frac{1}{N} \sum_{n=1}^N u_n * err(y_n, h(x_n))$
 - 可以增大稀有样本的权重，当分错稀有样本时产生很大的代价，从而平衡分类器的预测结果

分层抽样划分交叉验证集

Stratified Sampling Cross-Validation



分层抽样划分交叉验证集

Stratified Sampling Cross-Validation

不分层划分后CV内获奖比例不一致

CV0 (155 : 101 : 53)

CV1 (145 : 94 : 91)

CV2 (143 : 89 : 63)

CV3 (166 : 87 : 86)

CV4 (132 : 94 : 58)

获1000 : 获1500 : 获2000

训练集
(741 : 465 : 354)

要点：

1. 分层抽样可以大大提高CV的稳定性
2. 尝试不同的随机种子进行CV划分，选择与线上最为一致的CV

分层划分后CV内获奖比例一致


CV0 (149 : 94 : 71)

CV1 (145 : 94 : 71)

CV2 (149 : 89 : 71)

CV3 (149 : 94 : 70)

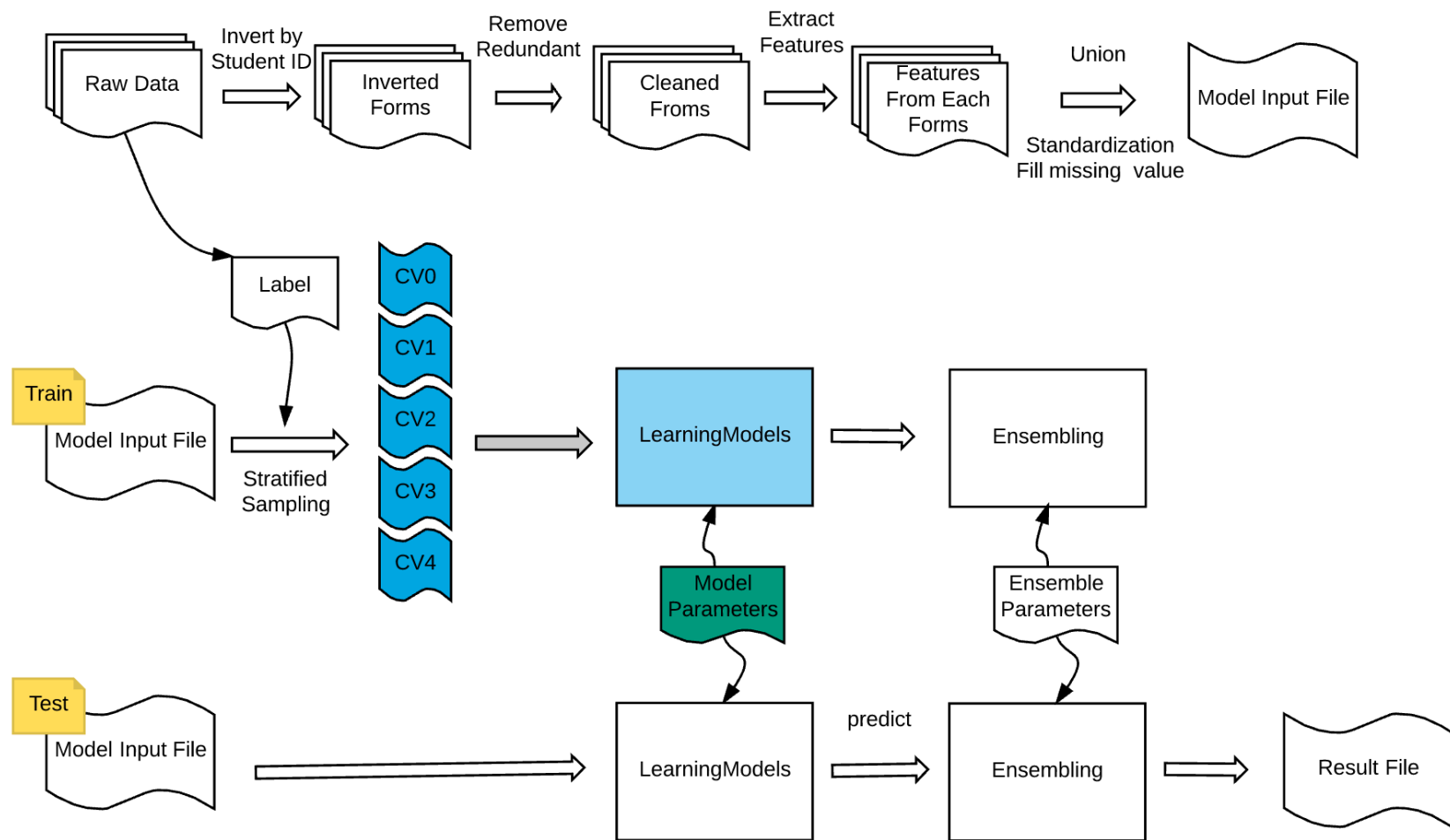
CV4 (149 : 94 : 70)



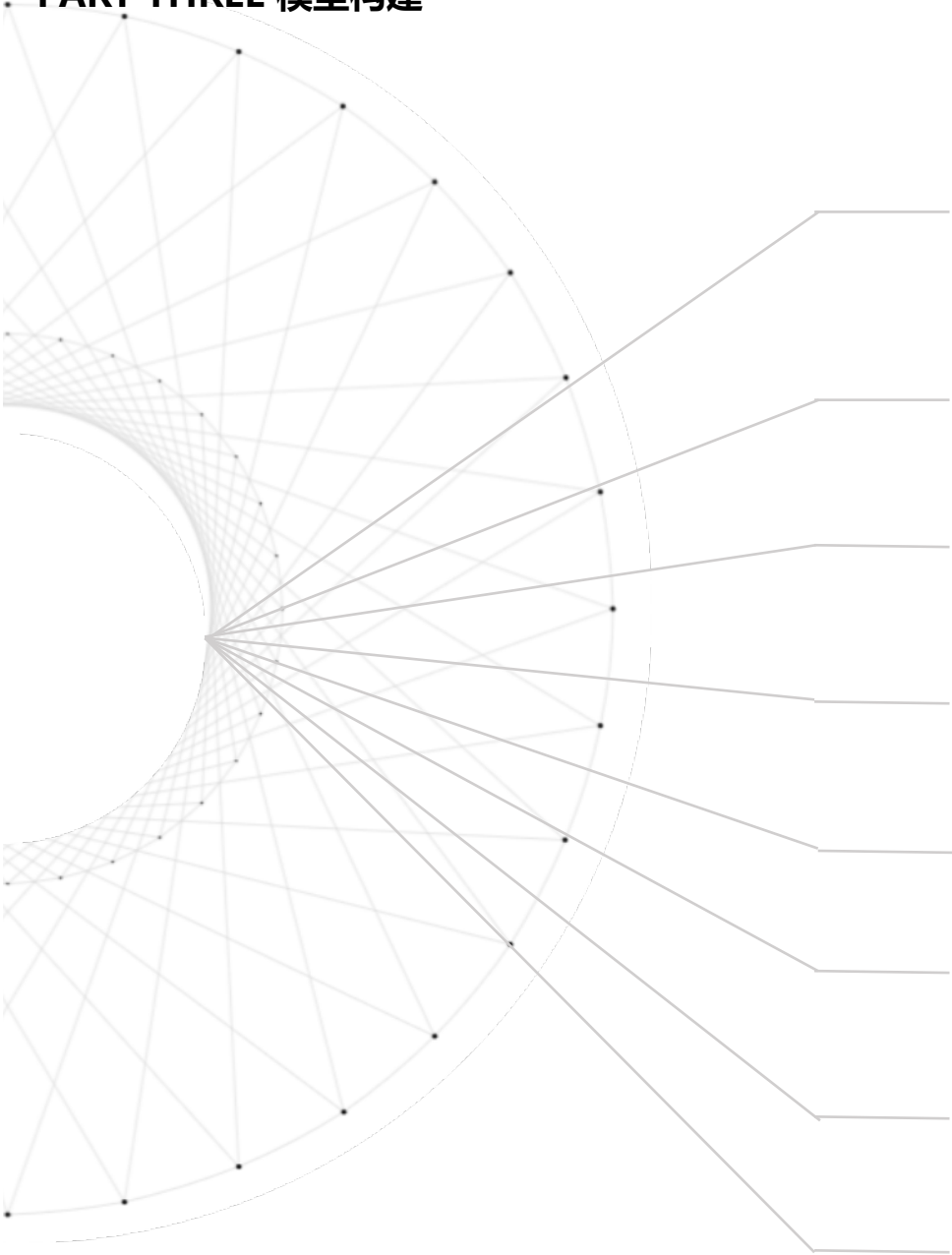
模型构建

PART THREE

PART THREE 模型构建



PART THREE 模型构建



- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

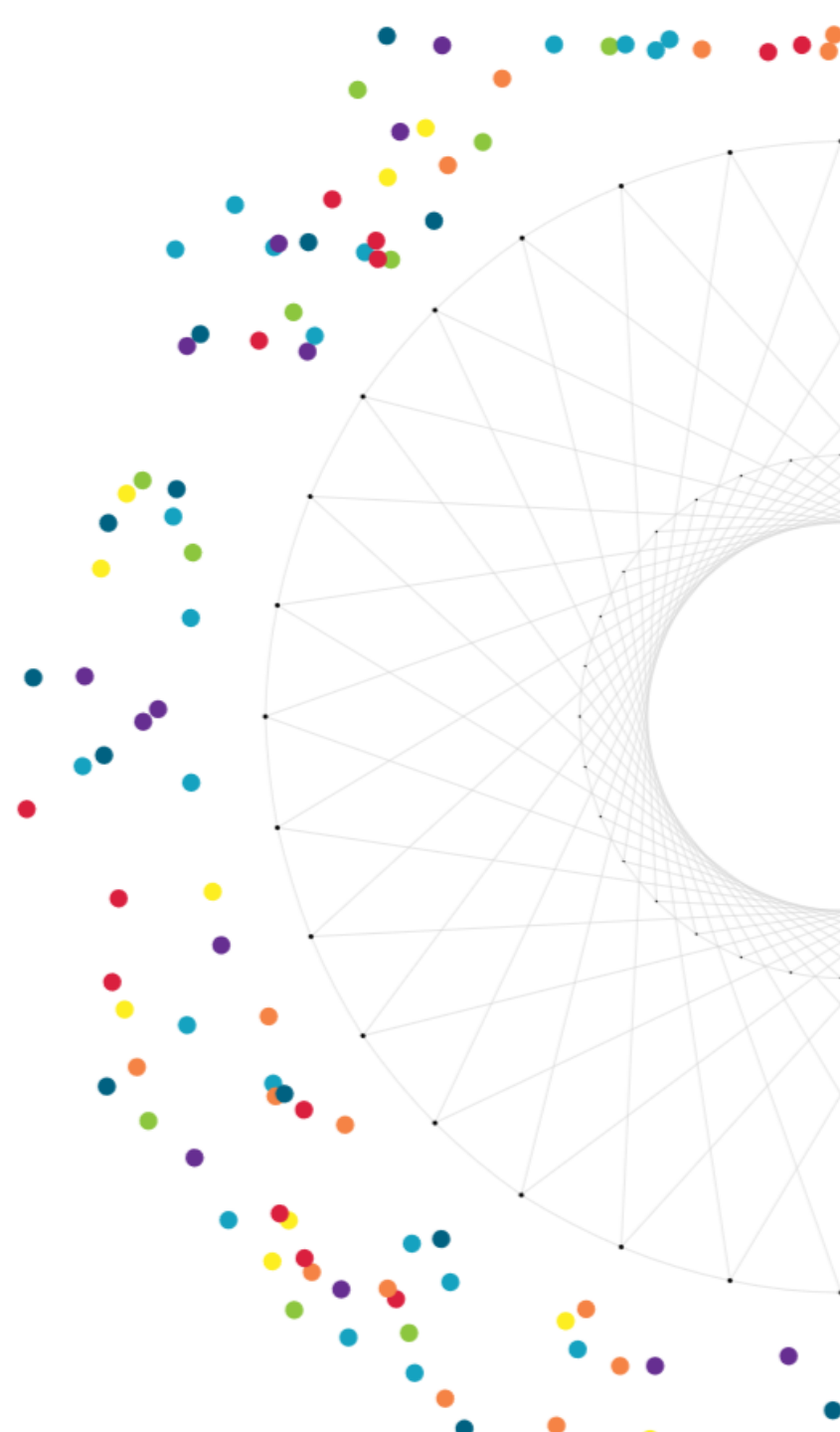
算法名		线上得分 (13前榜单)
Gradient Boosting Decision Tree (GDBT)		0.02889
Extreme Gradient Boosting (XGB)		>0.02830 <0.02889
Random Forest (RF)		>0.02830 <0.02889
Extremely Randomized Tree(ET)		>0.02889 <0.03006
AdaBoost (Ada)		0.03006
Support Vector Machine (SVM)		<0.02660
Shallow Neural Network(NN)		<0.02660
K Nearest Neighbors(KNN)		<0.02660

模型参数调优

Parameters Calibration

贪心坐标下降法：根据参数对模型的影响程度进行排序，然后按照重要性对参数依次进行优化

网格法：将参数空间分成网格，遍历所有网格中的点来寻找最优参数，比赛中，我们利用集群的优势使用了较细粒度的网格进行搜索。



GBDT 调参实例

GBDT 重要的参数有：

`sample_weight`(每个样本的权重，按照权重计算loss function)

`learning_rate` (学习率)

`n_estimators` (树的数量)

`max_depth`(单颗树高度)

`max_features`(分裂结点时考虑的特征数量)

`min_samples_leaf`(叶子节点要求的最小样本数)

`min_samples_split` (一个结点允许分裂的最小样本数)

调节`sample_weight`

在使用默认参数的情况下调节不同类别样本的权重，比如将不同类别权重设置为1:40:50:90

调节`n_estimators`

调节`n_estimators`使得模型达到一个较好的准确率

调节`max_depth` 和 `min_samples_leaf`

使用网格法调整以上两个参数, 实际上`min_samples_leaf`和`min_samples_split`起到的作用相同, 都是防止过拟合, 这两个的顺序可以调换

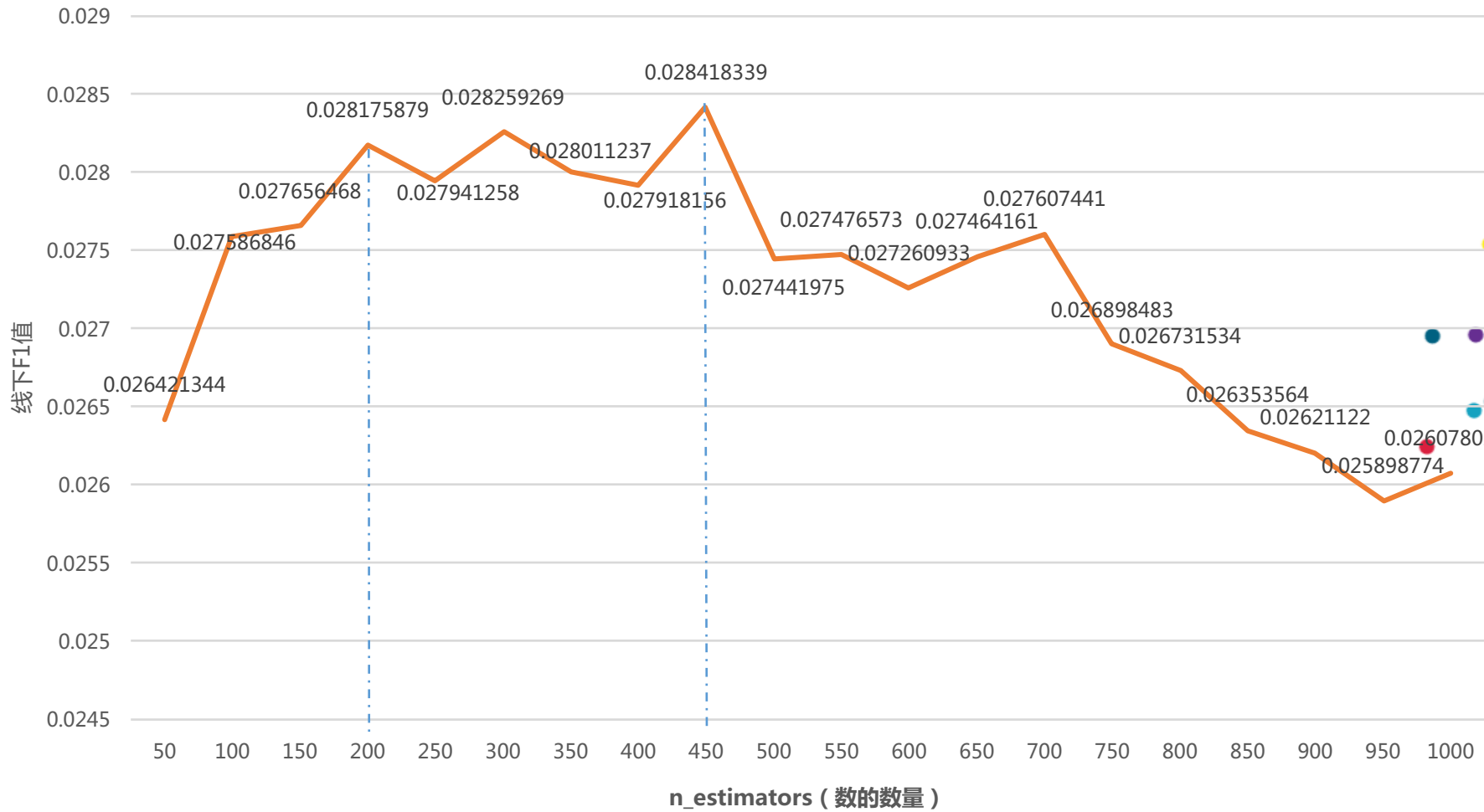
调节`learning_rate`和 `n_estimators`

使用网格法调整以上两个参数

调节`max_features`

调节`min_samples_split`

GBDT 调参实例

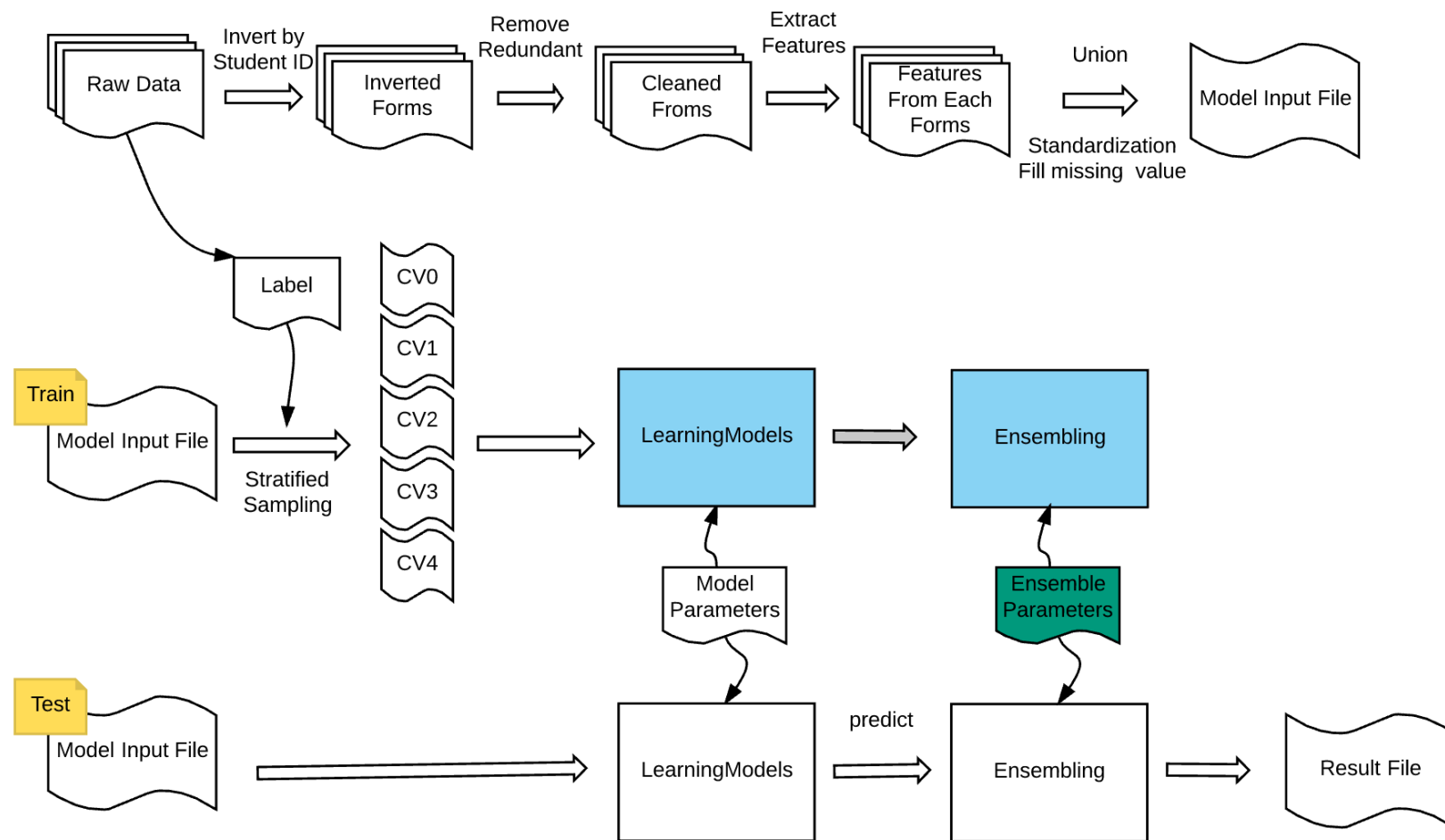




模型融合

PART FOUR

模型融合 Ensembling



单个算法拓展

使用不同的随机种子

同一种算法使用不同的随机种子将会拓展出很多不同的base learner。举例来说，在GDBT中随机种子会影响到单颗树考察的样本以及分裂结点时可选择的特征

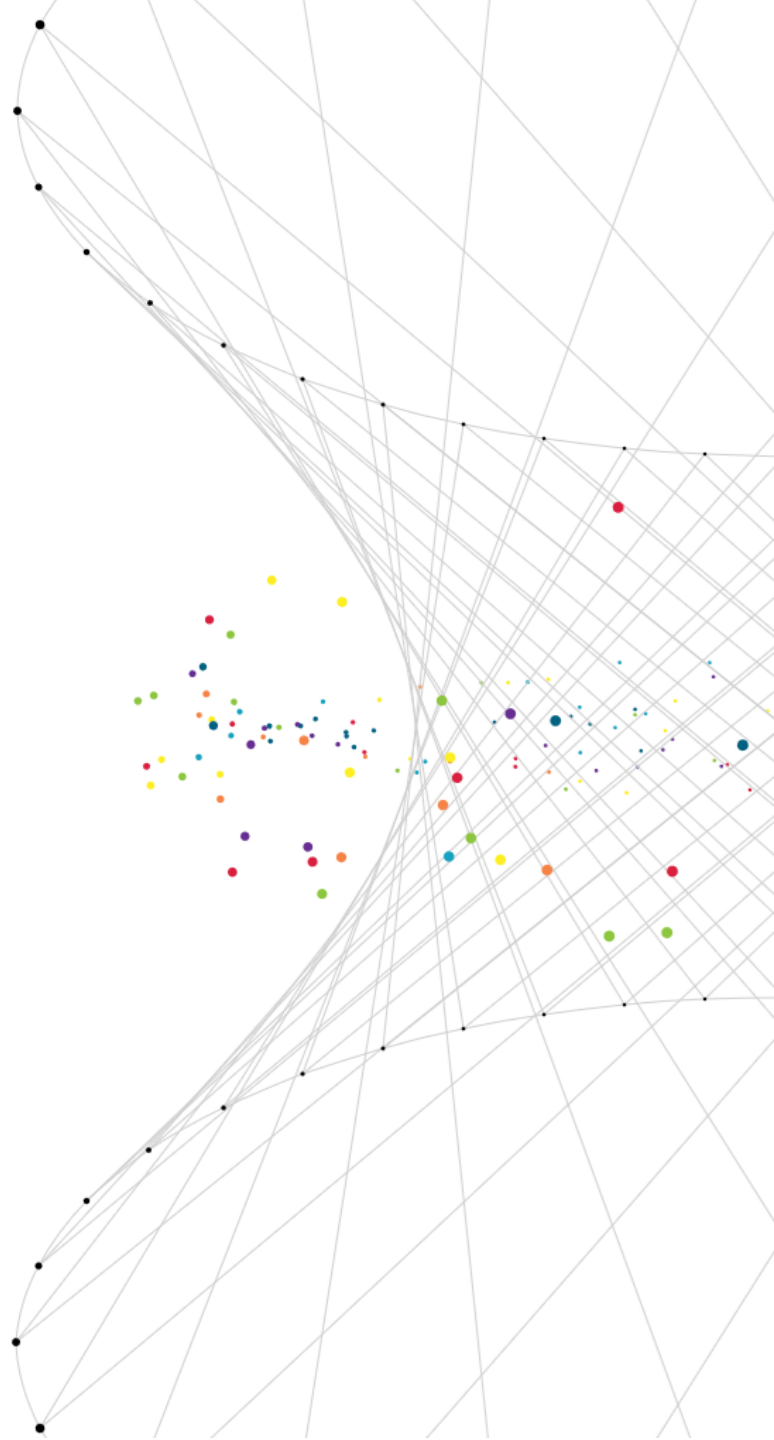
使用不同的预测比例

实验发现几种不同的预测比例都可以很高的精确度。举例来说，在GDBT中通过调节样本权重可以获得不同的预测比例，试验发现 不获奖：1000: 1500: 2000 的预测比例为以下三种时都可以获得很高的F1值：

- 1、7635:2124:634:550
- 2、7808:2117:677:359
- 3、8239:1929:484:309

使用不同的特征子集

根据所有的特征的重要性排序。根据这个重要性从1开始编号，按照 $\text{idx} \% 5$ 把所有feature划分成5份；同时再随机产生5份 feature 子集。这10份feature 子集每一份都可以训练一个模型



多个模型融合

Leavel0

GDBT 预测比例1 随
机种子1

GDBT 预测比例1 随
机种子2



GDBT 预测比例1 随
机种子30

GDBT 预测比例1 特
征子集1

GDBT 预测比例1 特
征子集2



GDBT 预测比例1 特
征子集30

Leavel1

VotingClassifier
(GDBT1)

VotingClassifier
(GDBT2)

VotingClassifier
(GDBT3)

VotingClassifier
(ET)

VotingClassifier
(Ada)

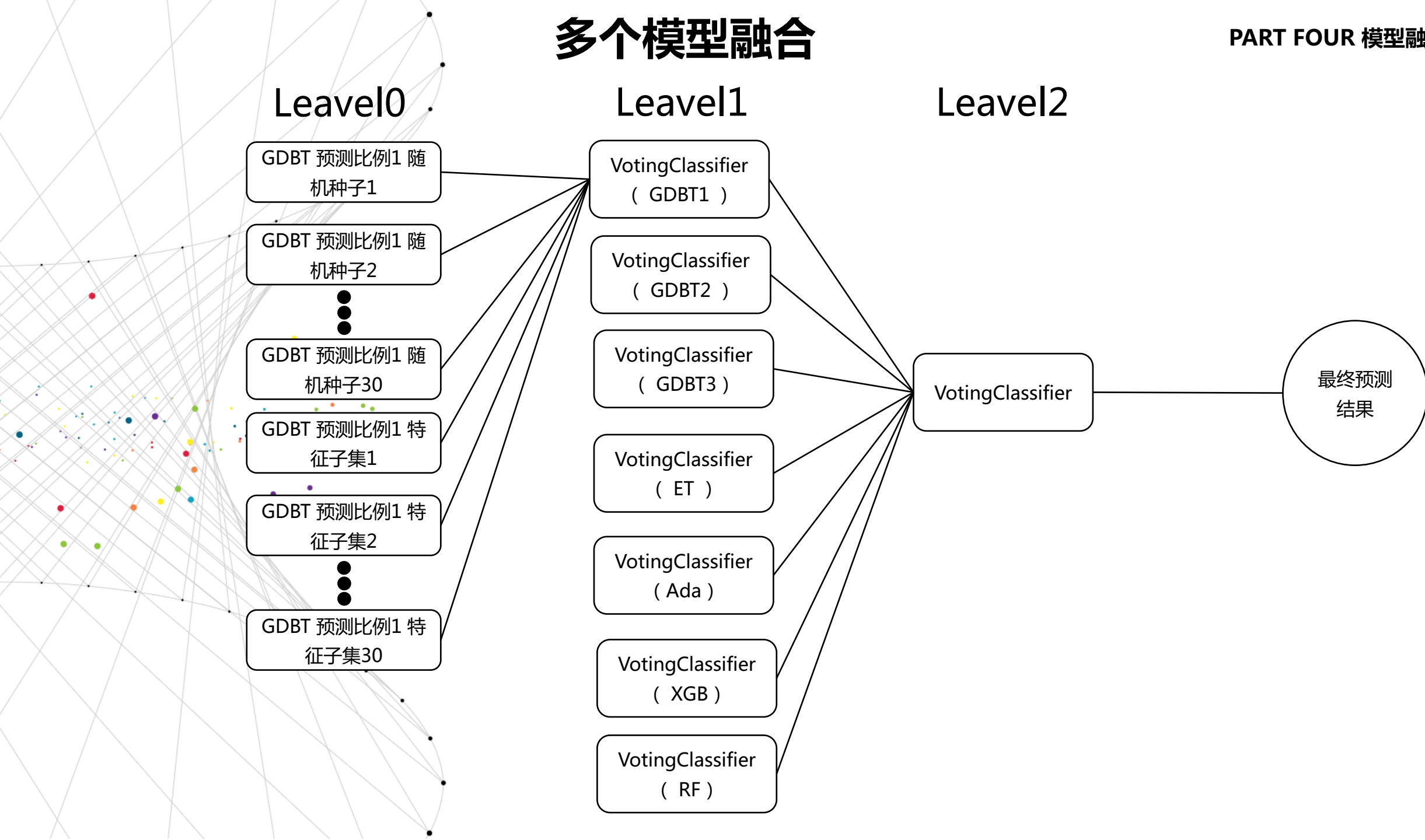
VotingClassifier
(XGB)

VotingClassifier
(RF)

Leavel2

VotingClassifier

最终预测
结果

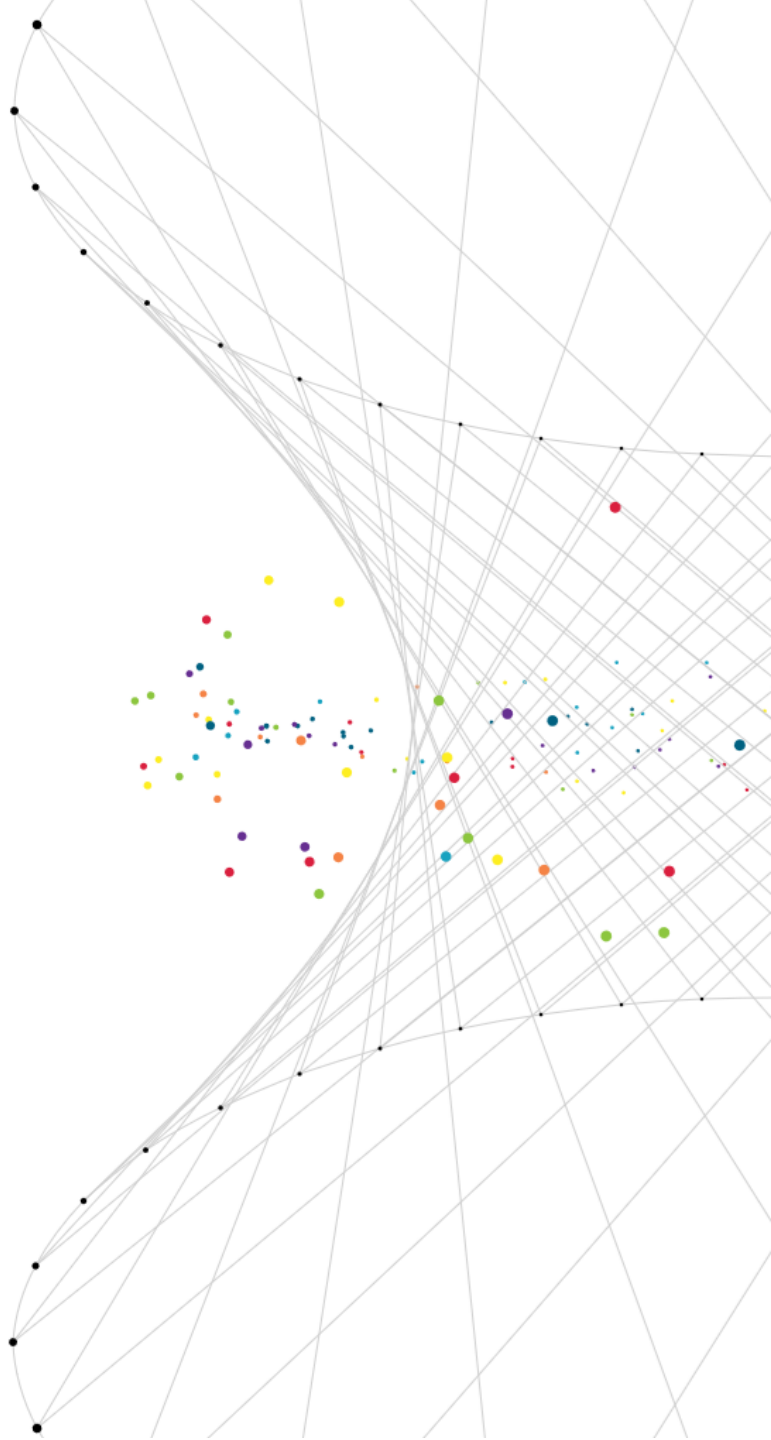


最佳融合权重

算法	GDBT1	GDBT2	GDBT3	XGB	Ada	RF	ET
比例	2	0.5	0.5	1.5	1.5	1	1

通过以上比例的融合，我们最终的线上得分为0.03143，相比于单个的GDBT模型提升了0.001。这使得我们在最后一天以很大的优势登上排行榜榜首的位置。

排名	排名变化	头像	队名	最高得分	提交次数	最后提交时间	创新应用简介
1	-		Yancy&Zhendong	0.03143	11	2017-02-20 23:50	已提交
2	-		Say what all late	0.03049	10	2017-02-17 03:39	未提交
3	-		贝叶斯部落	0.03049	12	2017-02-20 23:52	已提交
4	-		呵呵_000	0.02969	11	2017-02-20 23:58	已提交
5	-		liuyongkang_sysu4c134	0.02933	4	2017-02-17 20:11	未提交
6	-		top	0.02914	13	2017-02-21 01:08	未提交
7	-		皮皮虾我们走	0.02888	14	2017-02-20 23:59	已提交
8	-		calmy19946de52	0.02885	6	2017-02-16 17:53	未提交
9	-		qqE0A55A95B2D129FD4	0.02868	2	2017-02-17 17:27	未提交



感想与结论

PART FIVE

建立有效的线下评测机制

使用随机分层抽样的方法构建交叉验证集，尽可能的保证线下和线上分数同步

多层次融合

不同随机种子，不同特征子集，不同预测比例，不同模型多个层次的融合能够将融合的性能发挥到极致。

深入理解业务，完善的特征工程机制

调参只能小范围提高分数。要想大幅度的提升算法效果，需要深入分析业务领域，不放过任何学生的行为信息，细致的进行特征工程

细致的数据预处理

在构建模型之前，我们进行了完整的数据预处理流程：数据分析，数据清洗，数据归一化，缺失值填充。



制胜关键

PART FIVE 感想与结论

模型小评

本次比赛中，我队伍尝试了众多分类器，总的来说，采用树结构的分类模型要比非树形结构的分类器要稳定，效果更佳。Ensemble的方法要明显优于单个效果。

敏捷开发，迭代编码

先让你的代码把整个流程跑起来，再往里面添血加肉，逐步完善。划分好清晰的模块，尽可能提高代码的复用性、易用性。代码和数据要分离存放。可用git等工具提高代码管理效率。

不到最后一刻绝不放弃

要相信自己，只要方法正确，不停尝试，一定会有提高。特别是比赛的最后几天，其实才是最关键的时刻。要紧牙关到最后一刻。

正确使用随机种子

大部分模型的训练都是具有随机性的，虽然随机性可以提高效果，但是它给调参和debug带来了巨大挑战。在调试阶段一定要固定随机种子，是的每次随机运行的结果相同。

预测结果的分类比例至关重要

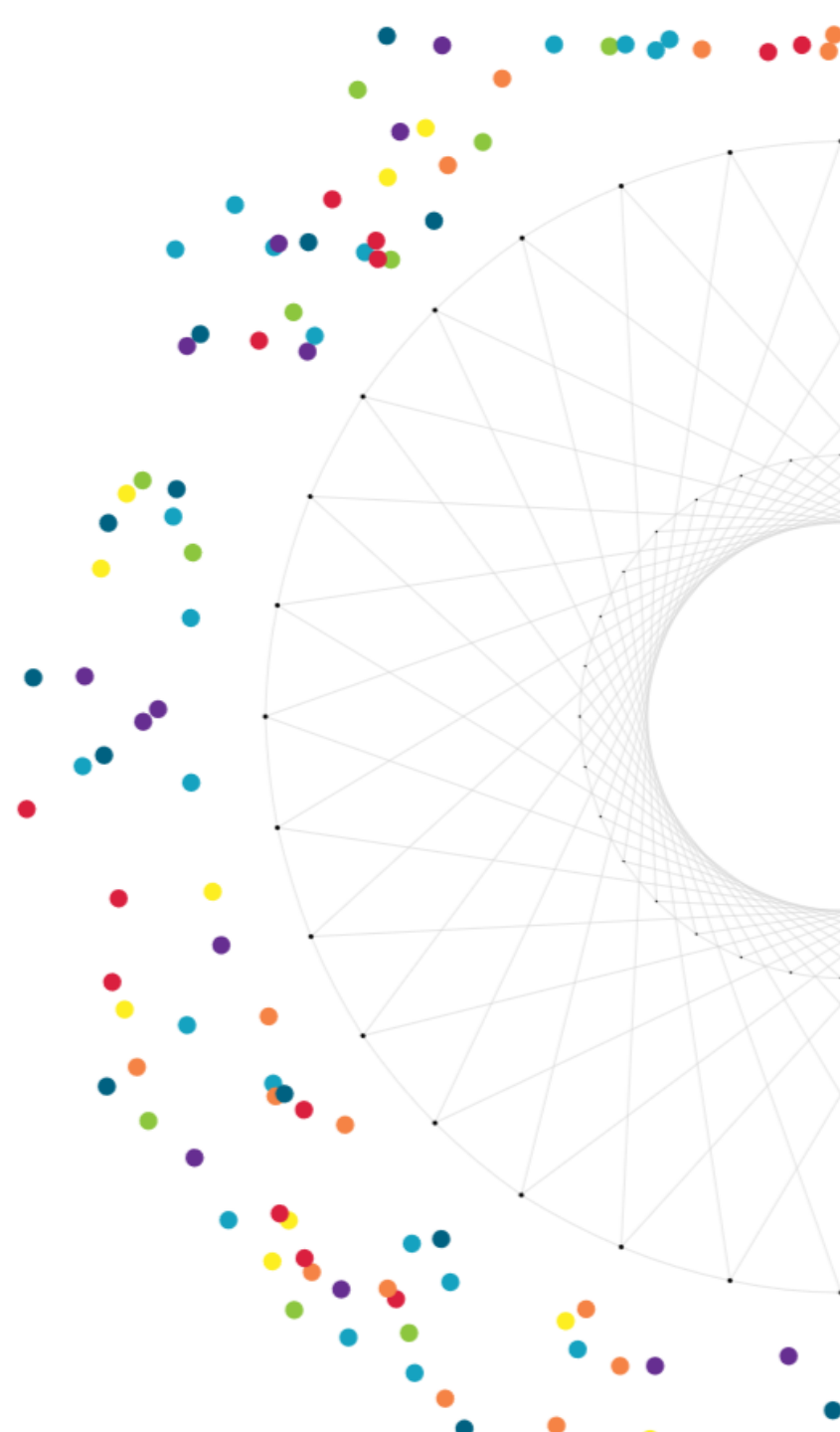
- 不在正确的比例下调参就相当于走入了岔路
- 使用不同的参数和特征会使最佳比例变化
- 最佳比例不止一个，一般来说是正确数目的两倍

放眼未来，不要计较一城一池的得失

- 如果很长一段时间调参，融合都没有很大提升，就要思考换一种方法，走另一条路，比如再新加一些特征，删除某些不好的特征，切换另一种数据预处理策略。不要执着与目前榜单被超过了多少名，榜单的变化往往是跳跃式的

参考文献

1. Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of Machine Learning Research* 12.Oct (2011): 2825-2830.
2. KAGGLE ENSEMBLING GUIDE (<http://mlwave.com/kaggle-ensembling-guide/>)
3. Dietterich, Thomas G. "Ensemble methods in machine learning." International workshop on multiple classifier systems. Springer Berlin Heidelberg, 2000.
4. de Abril, Ildefons Magrans, and Masashi Sugiyama. "Winning the kaggle algorithmic trading challenge with the composition of many models and feature engineering." *IEICE TRANSACTIONS on Information and Systems* 96.3 (2013): 742-745.
5. Taieb, Souhaib Ben, and Rob J. Hyndman. "A gradient boosting approach to the Kaggle load forecasting competition." *International Journal of Forecasting* 30.2 (2014): 382-394.
6. Puurula, Antti, Jesse Read, and Albert Bifet. "Kaggle LSHTC4 winning solution." arXiv preprint arXiv:1405.0546 (2014).





THANK YOU FOR WATCHING

PRESENTED BY Yancy&Zhendong