随机森林

基于R语言的组合方法

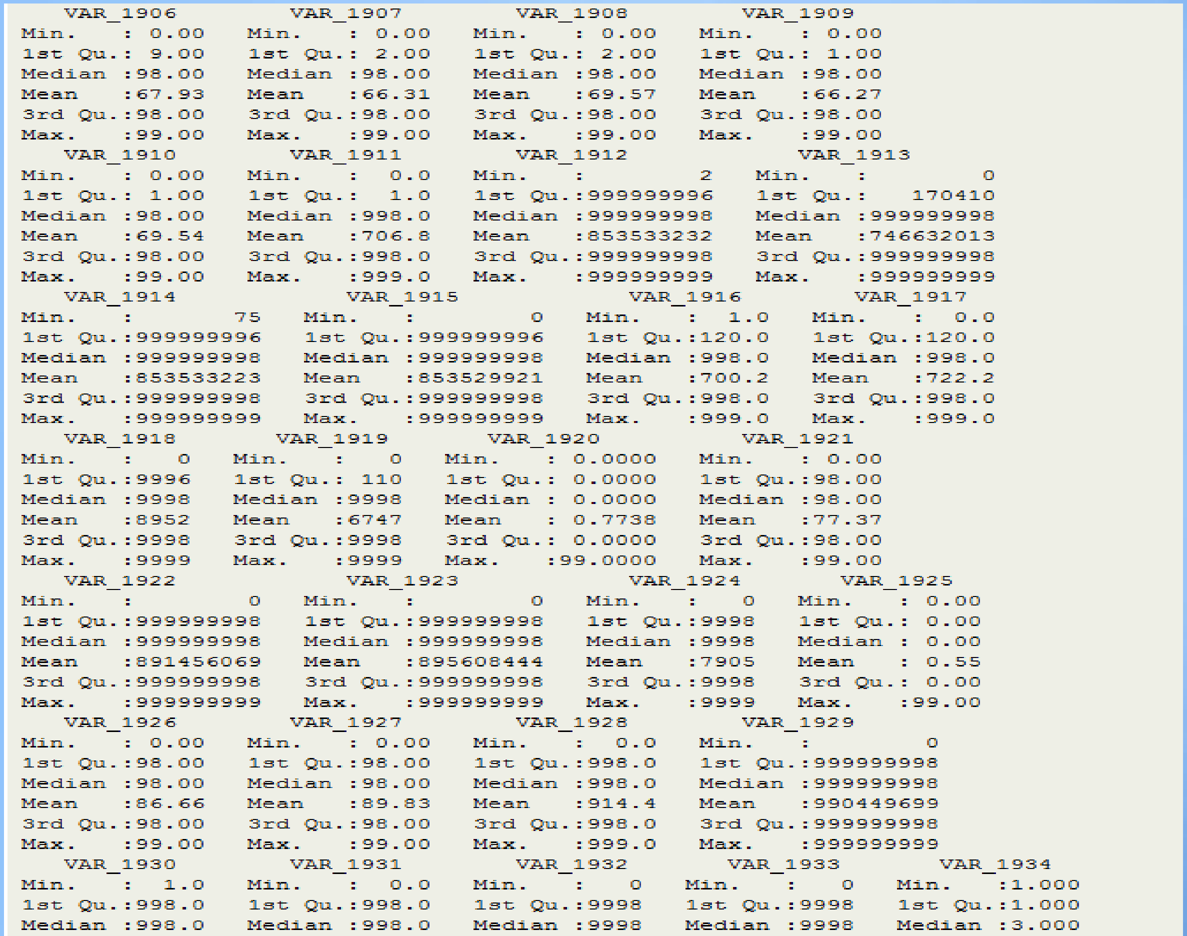
R语言简介

- 诞生于AT&T的贝尔实验室，是一种基于S语言的开源实现
- 超过5000个统计、机器学习、数据可视化、金融模型包
- 便捷的数据预处理和卓越的绘图功能
- SparkR和Rhadoop等开源项目支持；微软和Oracle等公司也有专门的R语言包

实验数据集

- 选择kaggle中springleaf公司提供数据集
- 训练数据和测试数据均为920M
- 数据145232行，属性数目1934列
- 每列熟悉数据均被隐去实际含义

实验数据集初探



```
   VAR_1906        VAR_1907        VAR_1908        VAR_1909
Min.   : 0.00   Min.   : 0.00   Min.   : 0.00   Min.   : 0.00
1st Qu.: 9.00   1st Qu.: 2.00   1st Qu.: 2.00   1st Qu.: 1.00
Median :98.00   Median :98.00   Median :98.00   Median :98.00
Mean   :67.93   Mean   :66.31   Mean   :69.57   Mean   :66.27
3rd Qu.:98.00   3rd Qu.:98.00   3rd Qu.:98.00   3rd Qu.:98.00
Max.   :99.00   Max.   :99.00   Max.   :99.00   Max.   :99.00
   VAR_1910        VAR_1911        VAR_1912            VAR_1913
Min.   : 0.00   Min.   :  0.0   Min.   :        2   Min.   :        0
1st Qu.: 1.00   1st Qu.:  1.0   1st Qu.:999999996   1st Qu.:   170410
Median :98.00   Median :998.0   Median :999999998   Median :999999998
Mean   :69.54   Mean   :706.8   Mean   :853533232   Mean   :746632013
3rd Qu.:98.00   3rd Qu.:998.0   3rd Qu.:999999998   3rd Qu.:999999998
Max.   :99.00   Max.   :999.0   Max.   :999999999   Max.   :999999999
   VAR_1914            VAR_1915            VAR_1916        VAR_1917
Min.   :       75   Min.   :        0   Min.   :  1.0   Min.   :  0.0
1st Qu.:999999996   1st Qu.:999999996   1st Qu.:120.0   1st Qu.:120.0
Median :999999998   Median :999999998   Median :998.0   Median :998.0
Mean   :853533223   Mean   :853529921   Mean   :700.2   Mean   :722.2
3rd Qu.:999999998   3rd Qu.:999999998   3rd Qu.:998.0   3rd Qu.:998.0
Max.   :999999999   Max.   :999999999   Max.   :999.0   Max.   :999.0
   VAR_1918       VAR_1919       VAR_1920          VAR_1921
Min.   :   0   Min.   :   0   Min.   : 0.0000   Min.   : 0.00
1st Qu.:9996   1st Qu.: 110   1st Qu.: 0.0000   1st Qu.:98.00
Median :9998   Median :9998   Median : 0.0000   Median :98.00
Mean   :8952   Mean   :6747   Mean   : 0.7738   Mean   :77.37
3rd Qu.:9998   3rd Qu.:9998   3rd Qu.: 0.0000   3rd Qu.:98.00
Max.   :9999   Max.   :9999   Max.   :99.0000   Max.   :99.00
   VAR_1922            VAR_1923            VAR_1924       VAR_1925
Min.   :        0   Min.   :        0   Min.   :   0   Min.   : 0.00
1st Qu.:999999998   1st Qu.:999999998   1st Qu.:9998   1st Qu.: 0.00
Median :999999998   Median :999999998   Median :9998   Median : 0.00
Mean   :891456069   Mean   :895608444   Mean   :7905   Mean   : 0.55
3rd Qu.:999999998   3rd Qu.:999999998   3rd Qu.:9998   3rd Qu.: 0.00
Max.   :999999999   Max.   :999999999   Max.   :9999   Max.   :99.00
   VAR_1926        VAR_1927        VAR_1928        VAR_1929
Min.   : 0.00   Min.   : 0.00   Min.   :  0.0   Min.   :        0
1st Qu.:98.00   1st Qu.:98.00   1st Qu.:998.0   1st Qu.:999999998
Median :98.00   Median :98.00   Median :998.0   Median :999999998
Mean   :86.66   Mean   :89.83   Mean   :914.4   Mean   :990449699
3rd Qu.:98.00   3rd Qu.:98.00   3rd Qu.:998.0   3rd Qu.:999999998
Max.   :99.00   Max.   :99.00   Max.   :999.0   Max.   :999999999
   VAR_1930        VAR_1931        VAR_1932       VAR_1933       VAR_1934
Min.   :  1.0   Min.   :  0.0   Min.   :   0   Min.   :   0   Min.   :1.000
1st Qu.:998.0   1st Qu.:998.0   1st Qu.:9998   1st Qu.:9998   1st Qu.:1.000
Median :998.0   Median :998.0   Median :9998   Median :9998   Median :3.000
```

数据预处理

- 对14W行数据进行采样

```
nsample=70000
```

- 非数值数据转化为

```
levels <- unique(c(train[[f]], test[层次变量[f]]))
```

- 数值类型进行标准化 $x = \frac{x-\mu}{\sigma}$

```
train[[f]]=scale(train[[f]],center=TRUE)
```

- 用-1替代空值（层次变量）

随机森林算法

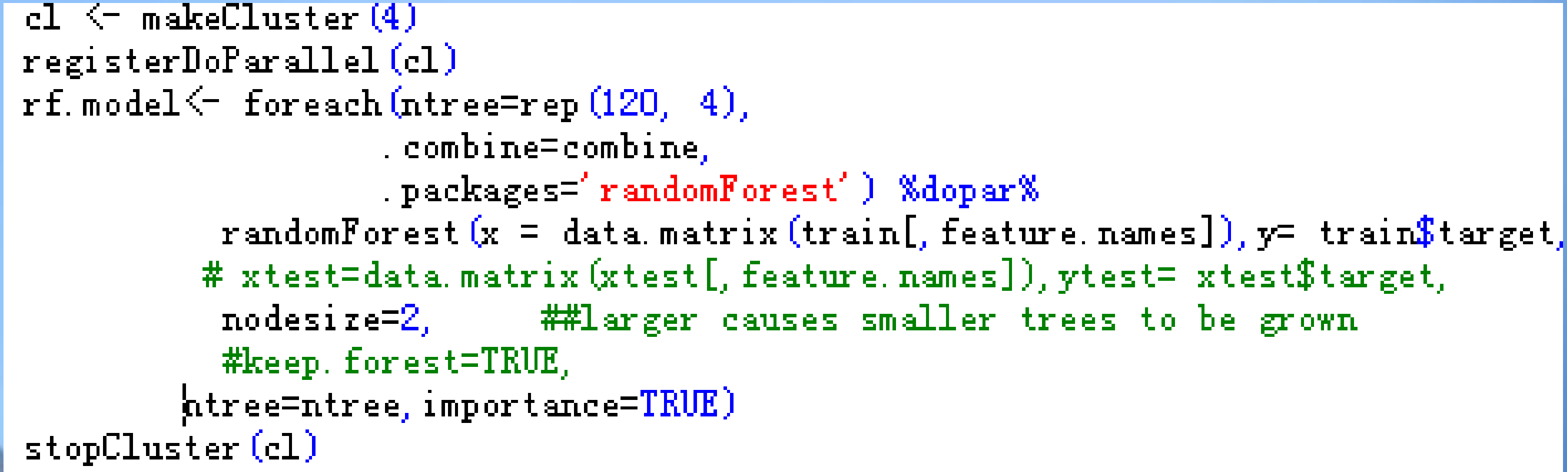
- 随机森林是基于决策树设计的组合方法
- 有放回进行采样、随机选择属性、无剪枝构建单个弱分类器
- 基于多个弱分类器投票构建模型组合形成强分类器，并且模型不易过拟合
- 模型训练可以并行化计算，加速建模过程

随机森林参数设置

并行算法，构建480棵决策树

依据属性的重要性进行模型调整优化

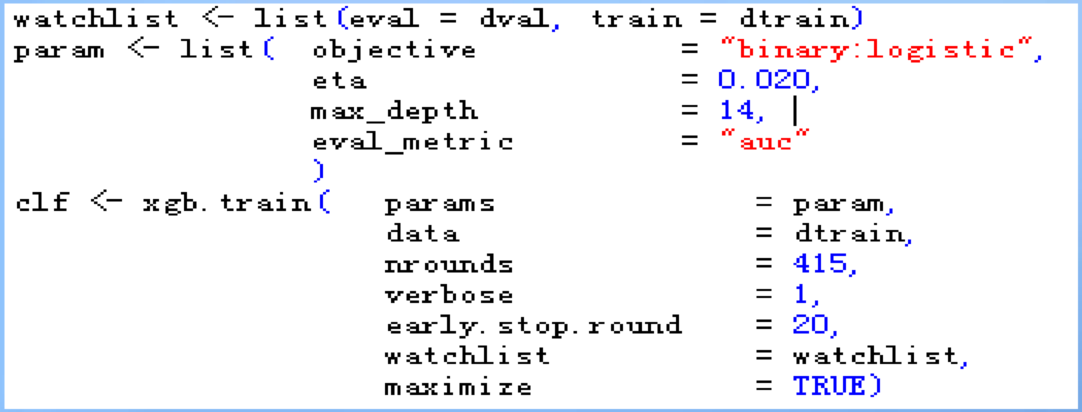
机器计算能力受限结果预测准确率约为0.75



```
cl <- makeCluster(4)
registerDoParallel(cl)
rf.model<- foreach(ntree=rep(120, 4),
                   .combine=combine,
                   .packages='randomForest') %dopar%
          randomForest(x = data.matrix(train[,feature.names]),y= train$target,
         # xtest=data.matrix(xtest[,feature.names]),ytest= xtest$target,
          nodesize=2,      ##larger causes smaller trees to be grown
          #keep.forest=TRUE,
        ntree=ntree,importance=TRUE)
stopCluster(cl)
```

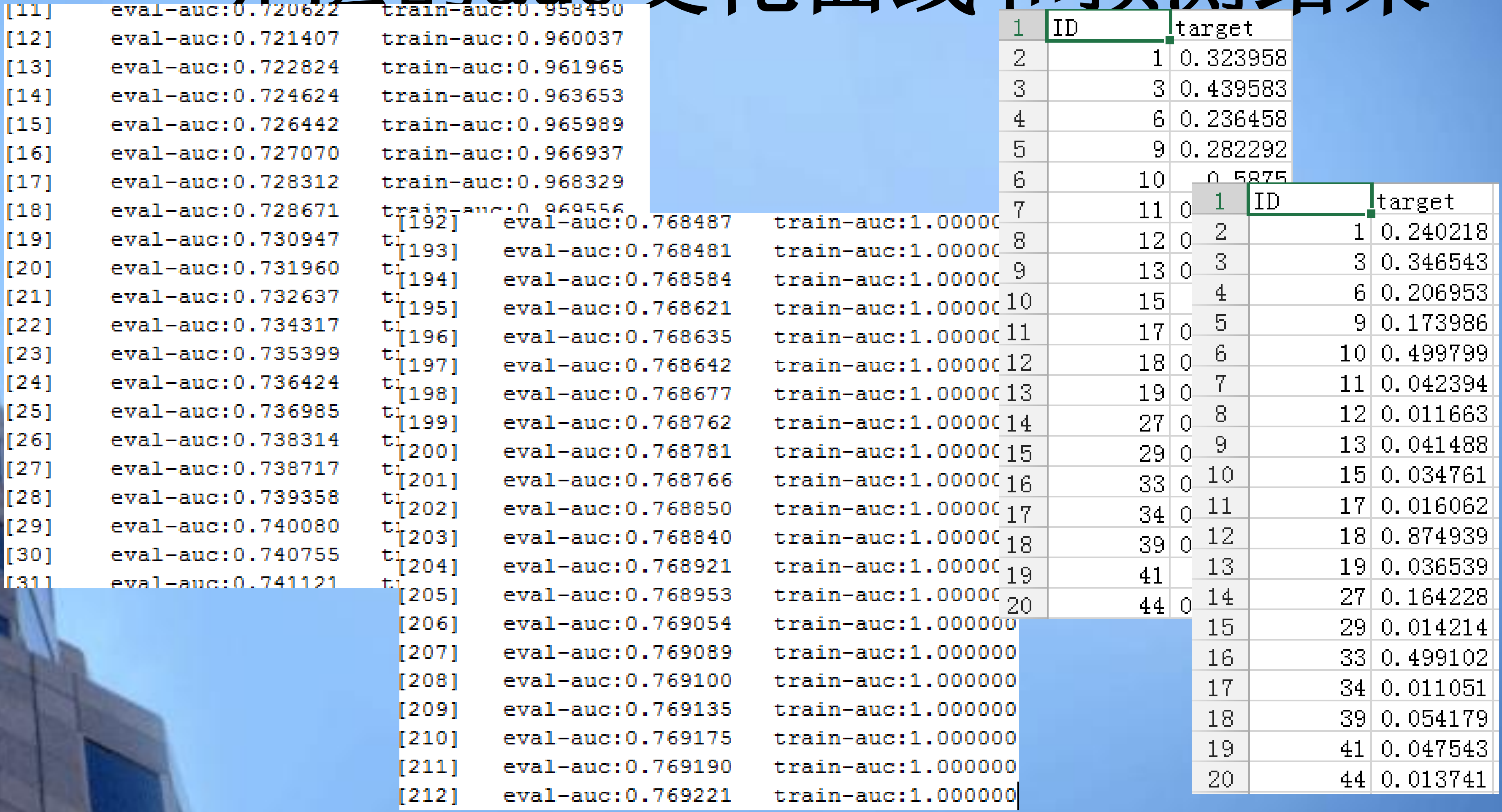
结合xgboost的pipline算法

- 训练模型中将数据集分为训练集和测试集，对模型进行验证并提高准确率
- 并行计算和逻辑聚类结合，通过随机梯度下降的方法提高模型的分类准确率
- 基于ROC和AUC度量模型评价其预测能力



```
watchlist <- list(eval = dval, train = dtrain)
param <- list(  objective           = "binary:logistic",
                eta                 = 0.020,
                max_depth           = 14,
                eval_metric         = "auc"
                )
clf <- xgb.train(   params                 = param,
                    data                   = dtrain,
                    nrounds                = 415,
                    verbose                = 1,
                    early.stop.round       = 20,
                    watchlist              = watchlist,
                    maximize               = TRUE)
```

算法的auc变化曲线和预测结果



```
[11]    eval-auc:0.720622   train-auc:0.958450
[12]    eval-auc:0.721407   train-auc:0.960037
[13]    eval-auc:0.722824   train-auc:0.961965
[14]    eval-auc:0.724624   train-auc:0.963653
[15]    eval-auc:0.726442   train-auc:0.965989
[16]    eval-auc:0.727070   train-auc:0.966937
[17]    eval-auc:0.728312   train-auc:0.968329
[18]    eval-auc:0.728671   train-auc:0.969556
[19]    eval-auc:0.730947   t
[20]    eval-auc:0.731960   t
[21]    eval-auc:0.732637   t
[22]    eval-auc:0.734317   t
[23]    eval-auc:0.735399   t
[24]    eval-auc:0.736424   t
[25]    eval-auc:0.736985   t
[26]    eval-auc:0.738314   t
[27]    eval-auc:0.738717   t
[28]    eval-auc:0.739358   t
[29]    eval-auc:0.740080   t
[30]    eval-auc:0.740755   t
[31]    eval-auc:0.741121   t
```

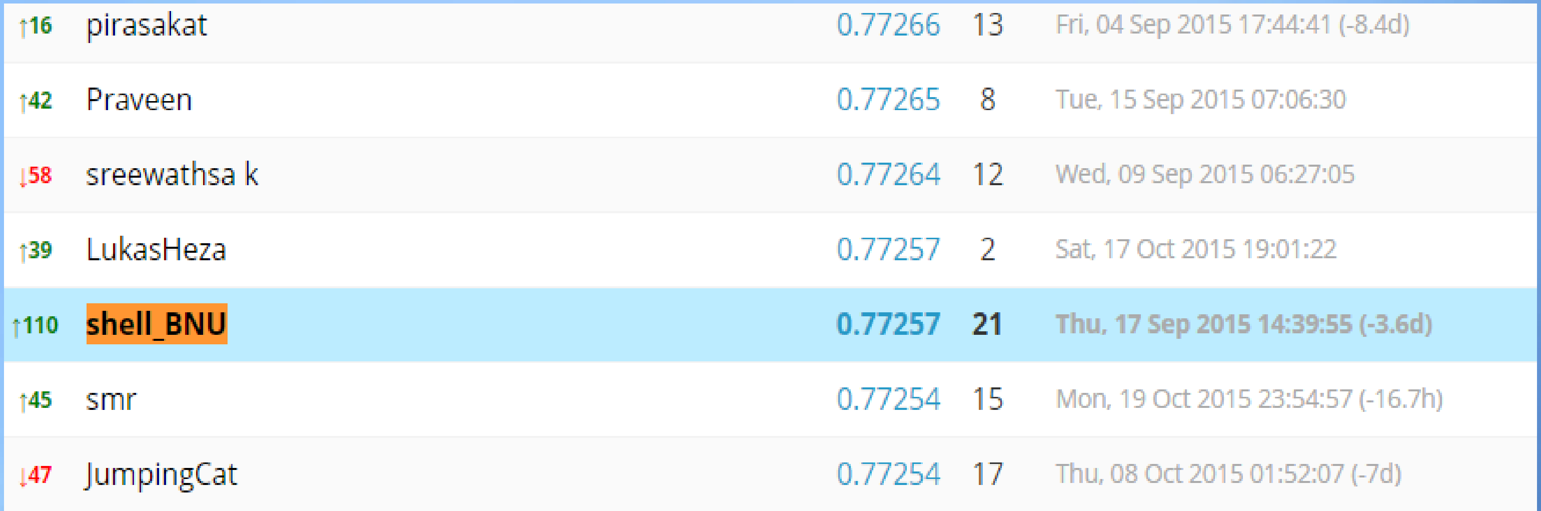
```
[192]   eval-auc:0.768487   train-auc:1.00000
[193]   eval-auc:0.768481   train-auc:1.00000
[194]   eval-auc:0.768584   train-auc:1.00000
[195]   eval-auc:0.768621   train-auc:1.00000
[196]   eval-auc:0.768635   train-auc:1.00000
[197]   eval-auc:0.768642   train-auc:1.00000
[198]   eval-auc:0.768677   train-auc:1.00000
[199]   eval-auc:0.768762   train-auc:1.00000
[200]   eval-auc:0.768781   train-auc:1.00000
[201]   eval-auc:0.768766   train-auc:1.00000
[202]   eval-auc:0.768850   train-auc:1.00000
[203]   eval-auc:0.768840   train-auc:1.00000
[204]   eval-auc:0.768921   train-auc:1.00000
[205]   eval-auc:0.768953   train-auc:1.00000
[206]   eval-auc:0.769054   train-auc:1.000000
[207]   eval-auc:0.769089   train-auc:1.000000
[208]   eval-auc:0.769100   train-auc:1.000000
[209]   eval-auc:0.769135   train-auc:1.000000
[210]   eval-auc:0.769175   train-auc:1.000000
[211]   eval-auc:0.769190   train-auc:1.000000
[212]   eval-auc:0.769221   train-auc:1.000000
```

1	ID	target
2	1	0.323958
3	3	0.439583
4	6	0.236458
5	9	0.282292
6	10	0.5875
7	11	0
8	12	0
9	13	0
10	15	
11	17	0
12	18	0
13	19	0
14	27	0
15	29	0
16	33	0
17	34	0
18	39	0
19	41	
20	44	0

1	ID	target
2	1	0.240218
3	3	0.346543
4	6	0.206953
5	9	0.173986
6	10	0.499799
7	11	0.042394
8	12	0.011663
9	13	0.041488
10	15	0.034761
11	17	0.016062
12	18	0.874939
13	19	0.036539
14	27	0.164228
15	29	0.014214
16	33	0.499102
17	34	0.011051
18	39	0.054179
19	41	0.047543
20	44	0.013741

Pipline结果分析

- 结合xgboost和随机森林方法，提高准确率到0.77，相当于多分类正确3000行数据
- 共提交21次结果，最好结果为0.77257



↑16	pirasakat	0.77266	13	Fri, 04 Sep 2015 17:44:41 (-8.4d)
↑42	Praveen	0.77265	8	Tue, 15 Sep 2015 07:06:30
↓58	sreewathsa k	0.77264	12	Wed, 09 Sep 2015 06:27:05
↑39	LukasHeza	0.77257	2	Sat, 17 Oct 2015 19:01:22
↑110	**shell_BNU**	**0.77257**	**21**	**Thu, 17 Sep 2015 14:39:55 (-3.6d)**
↑45	smr	0.77254	15	Mon, 19 Oct 2015 23:54:57 (-16.7h)
↓47	JumpingCat	0.77254	17	Thu, 08 Oct 2015 01:52:07 (-7d)

随机森林和xgboost的pipeline算法

黄勇　201521210022