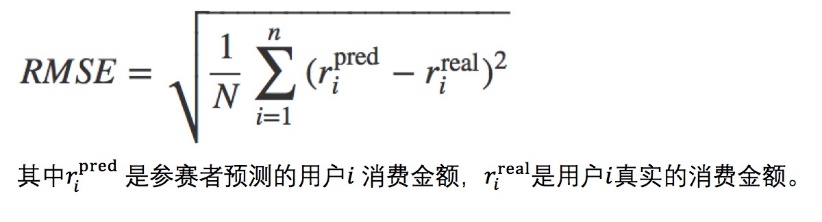
游戏玩家付费金额预测大赛

<http://www.dcjingsai.com/common/cmpt/%E6%B8%B8%E6%88%8F%E7%8E%A9%E5%AE%B6%E4%BB%98%E8%B4%B9%E9%87%91%E9%A2%9D%E9%A2%84%E6%B5%8B%E5%A4%A7%E8%B5%9B_%E7%AB%9E%E8%B5%9B%E4%BF%A1%E6%81%AF.html>

当你选择了一个比赛后，可以先“热热身”，稍微熟悉一下数据，粗略地跑出一些简单的模型，看看自己在榜上的排名，然后再去慢慢迭代。

评分标准 RMSE(root-mean-square error)，即均方根误差。



## 数据预处理

### 先取部分数据，用于快速实验

1减法train=0.1.csv ：训练集的1/10，样本数= 228800

2变换train=0.1-fe.csv ：train=0.1.csv 特征工程后的，注册日期转换为天数

3变换tap\_fun\_test-fe.csv ：tap\_fun\_test.csv 特征工程后的，注册日期转换为天数

4加法train=0.1-fe-top10.csv ：train=0.1-fe.csv 添加重要特征（top 10）的多项式

5加法tap\_fun\_test-fe-top10.csv ：tap\_fun\_test-fe.csv 添加重要特征（top 10）的多项式

6加法train=0.1-fe-top20.csv ：train=0.1-fe.csv 添加重要特征（top 20）的多项式

7加法tap\_fun\_test-fe-top20.csv ：tap\_fun\_test-fe.csv 添加重要特征（top 20）的多项式

8减法train=0.1-fe-coor0.3.csv ：train=0.1-fe.csv 取相关性 > 0.3的特征47个

9减法tap\_fun\_test-fe-coor0.3.csv ：tap\_fun\_test-fe.csv 取相关性 > 0.3的特征47个

10减法train=0.1-fe-coor0.3-pay.csv ：train=0.1-fe-coor0.3.csv 去掉不付费的记录，只保留付费记录

11减法train=0.1-fe-pay.csv ：train=0.1-fe.csv 去掉不付费的记录，只保留付费记录

### 样本平衡问题

利用EasyEnsemble方法，产生多个不同的训练集，训练多个分类器，组合多个分类器的结果得到最终结果。

由于只有1/50的玩家付费=正例，所以把数据集分成50份，每份包含所有的正例和1/50反例。

每个数据集，采用5折交叉验证。

train=0.1-01.csv ~ train=0.1-50.csv

### 最后用全部数据，最佳策略训练

tap\_fun\_train.csv ：整个训练集，样本数= 2288004

## 模型融合

我们使用 greedy blending 策略和两层结合了其他的图像特征的集成学习算法将模型结果融合起来。

确保所有的模型都要经过4折交叉验证，尽管这会增加计算开销，但是却能降低过拟合的风险。使用固定的k-fold 避免Valid Set过拟合。

### 基础学习器

复杂模型提取特征

#### 线性回归

#### GBDT+FFM

#### DeepFM

### 次级学习器

#### 1．SVM

#### 2．LightGBM

#### 3．Xgboost

### 最终结果：加权平均

# 日程记录

## 2018.06.24 数据探索

### 各个特征的含义

tap4fun数据字段-分析.xls

### 特征之间的相关性

# 升序排列，取 prediction\_pay\_price 相关性 > 0.3 有 48 个特征

| 特征名称 | **pay\_price** | **pay\_count** | **prediction\_pay\_price** |
| --- | --- | --- | --- |
| **sr\_infantry\_atk\_level** | 0.350232 | 0.656094 | 0.320464 |
| **magic\_add\_value** | 0.401094 | 0.349691 | 0.328770 |
| **cavalry\_add\_value** | 0.390930 | 0.538951 | 0.333436 |
| **sr\_rss\_e\_gather\_level** | 0.412443 | 0.431104 | 0.335987 |
| **sr\_cavalry\_atk\_level** | 0.372536 | 0.667096 | 0.341669 |
| **sr\_march\_size\_level** | 0.489823 | 0.204157 | 0.344306 |
| **sr\_outpost\_tier\_4\_level** | 0.489823 | 0.204157 | 0.344306 |
| **sr\_guest\_troop\_capacity\_level** | 0.489823 | 0.204157 | 0.344306 |
| **sr\_shaman\_atk\_level** | 0.387948 | 0.657870 | 0.354319 |
| **sr\_outpost\_tier\_3\_level** | 0.389628 | 0.530950 | 0.390737 |
| **sr\_troop\_attack\_level** | 0.599588 | 0.268091 | 0.403624 |
| **shaman\_add\_value** | 0.526324 | 0.557001 | 0.409337 |
| **infantry\_add\_value** | 0.470065 | 0.564580 | 0.409438 |
| **sr\_gathering\_march\_speed\_level** | 0.564067 | 0.403341 | 0.435076 |
| **sr\_pvp\_march\_speed\_level** | 0.570141 | 0.416220 | 0.442595 |
| **sr\_shaman\_tier\_4\_level** | 0.687182 | 0.333949 | 0.448995 |
| **sr\_alliance\_march\_speed\_level** | 0.617740 | 0.414411 | 0.455331 |
| **sr\_cavalry\_tier\_4\_level** | 0.739613 | 0.383548 | 0.511766 |
| **sr\_cavalry\_tier\_3\_level** | 0.603622 | 0.452134 | 0.522416 |
| **sr\_infantry\_tier\_3\_level** | 0.593288 | 0.453258 | 0.535145 |
| **sr\_shaman\_tier\_3\_level** | 0.595050 | 0.457640 | 0.541889 |
| **sr\_infantry\_tier\_4\_level** | 0.782337 | 0.404025 | 0.563373 |
| **building\_acceleration\_add\_value** | 0.680214 | 0.797693 | 0.589272 |
| **pay\_count** | 0.703309 | 1.000000 | 0.598916 |
| **sr\_troop\_defense\_level** | 0.704995 | 0.425913 | 0.607208 |
| **sr\_cavalry\_hp\_level** | 0.704053 | 0.422663 | 0.607486 |
| **building\_acceleration\_reduce\_value** | 0.703206 | 0.784215 | 0.608038 |
| **sr\_shaman\_def\_level** | 0.758099 | 0.427712 | 0.614711 |
| **sr\_infantry\_def\_level** | 0.728575 | 0.409429 | 0.621730 |
| **sr\_shaman\_hp\_level** | 0.715633 | 0.414688 | 0.622085 |
| **sr\_cavalry\_def\_level** | 0.835937 | 0.451455 | 0.629569 |
| **meat\_add\_value** | 0.792775 | 0.623492 | 0.650798 |
| **reaserch\_acceleration\_add\_value** | 0.798939 | 0.830616 | 0.671408 |
| **sr\_infantry\_hp\_level** | 0.814188 | 0.474725 | 0.681264 |
| **training\_acceleration\_add\_value** | 0.815864 | 0.743238 | 0.686599 |
| **reaserch\_acceleration\_reduce\_value** | 0.834280 | 0.782629 | 0.695218 |
| **meat\_reduce\_value** | 0.858870 | 0.713156 | 0.700602 |
| **training\_acceleration\_reduce\_value** | 0.844732 | 0.704375 | 0.712416 |
| **wood\_reduce\_value** | 0.870434 | 0.725854 | 0.712524 |
| **general\_acceleration\_add\_value** | 0.886887 | 0.793226 | 0.725020 |
| **wood\_add\_value** | 0.891059 | 0.684707 | 0.732175 |
| **stone\_add\_value** | 0.905230 | 0.634133 | 0.737585 |
| **general\_acceleration\_reduce\_value** | 0.895368 | 0.762830 | 0.740558 |
| **ivory\_reduce\_value** | 0.911758 | 0.603684 | 0.743139 |
| **ivory\_add\_value** | 0.909818 | 0.643655 | 0.746421 |
| **stone\_reduce\_value** | 0.924619 | 0.672541 | 0.747822 |
| **pay\_price** | 1.000000 | 0.703309 | 0.803923 |
| **prediction\_pay\_price** | 0.803923 | 0.598916 | 1.000000 |

### 数据集的特别之处

类似 CTR 数据集。

* 1 数据量大，数据非常稀疏。
* 2 负样本数目 >> 正样本数目

样本总量 228800

pay\_count 属性的不同取值和出现的次数

0 224599 一次都没有付费的最多，占 98.16%

1 1786

2 792

3 491

4 309

5 209

6 159

7 100

8 77

9 57

11 39

10 35

如何解决样本不平衡问题？利用EasyEnsemble的方法，即多次下采样（放回采样，这样产生的训练集才相互独立）产生多个不同的训练集，进而训练多个不同的分类器，通过组合多个分类器的结果得到最终的结果。

* 3 所有特征分布都是长尾，付费低的多而集中，付费高的少而分散。

<http://wiki.mbalib.com/wiki/%E9%95%BF%E5%B0%BE%E7%90%86%E8%AE%BA>

“长尾”案例：Google、亚马逊，涓涓细流，汇聚成河。

只要存储和流通的渠道足够大，需求不旺或销量不佳的产品共同占据的市场份额，就可以和那些数量不多的热卖品所占据的市场份额相匹敌，甚至更大。

所有特征值，取倒数，只使用1/10000以上的。

干脆去掉7日不付费的记录。训练个模型看看。

* 4 sr\_rss\_help\_bonus\_level 这个特征值，全是 0

训练集里面只有3个玩家 sr\_rss\_help\_bonus\_level > 0 而且是付费的

测试集里面也有3个玩家 sr\_rss\_help\_bonus\_level > 0 应该是付费的

## 2018.06.25 快速实验，用部分数据

**先用最小的数据集训练，看看效果。再逐渐扩大，比较效果。**

1. train=0.1-fe-coor0.3-pay.csv ：只保留付费记录，

对应测试集 tap\_fun\_test-fe-coor0.3.csv

* 无论是否标准化 train\_y ，LinearRegression、岭回归得到的预测结果相同
* 标准化 train\_y 后，Lasso回归得到的预测结果更好。
* **结论：标准化 train\_y 更好。**

LinearRegression summit0.1-coor0.3-pay-Linear.csv **= 97.8753**

岭回归 summit0.1-coor0.3-pay-ridge.csv **= 86.01859**

Lasso回归 标准化train\_y summit0.1-coor0.3-pay-lasso.csv **= 82.12618**

Lasso回归 不标准化train\_y summit0.1-coor0.3-pay-lasso\_1.csv **=** **91.08**

LightGBM 默认参数 summit0.1-coor0.3-pay-lgbm\_1.csv =99.10\*\*\*4

参数调优后 summit0.1-coor0.3-pay-lgbm-2.csv = 91.57\*\*\*5

1. train=0.1-fe-coor0.3.csv ：取相关性 > 0.3的特征47个

对应测试集 tap\_fun\_test-fe-coor0.3.csv

Lasso回归 summit0.1-coor0.3-lasso-1.csv **= 78.14601**

岭回归 summit0.1-coor0.3-RidgeCV-1.csv **= 86.35\*\*9**

1. train=0.1-fe-pay.csv ：只保留付费记录

对应测试集 tap\_fun\_test-fe.csv

Lasso回归 summit=0.1-fe-pay-lasso-1.csv **= 82.14\*\*2**

岭回归 summit=0.1-fe-pay-RidgeCV-2.csv **= ？？**

1. train=0.1-fe.csv ：特征工程后的

对应测试集 tap\_fun\_test-fe.csv

Lasso回归 ？？.csv **= ？？**

岭回归 ？？.csv **= ？？**

1. train=0.1-fe-top10.csv ：添加重要特征（top 10）的多项式

对应测试集 tap\_fun\_test-fe-top10.csv

Lasso回归 ？？.csv **= ？？**

岭回归 ？？.csv **= ？？**

1. train=0.1-fe-top20.csv ：添加重要特征（top 20）的多项式

对应测试集 tap\_fun\_test-fe-top20.csv

Lasso回归 ？？.csv **= ？？**

岭回归 ？？.csv **= ？？**

# 训练策略

## 快速实验，用部分数据

寻找最佳训练策略

## 训练基本学习器，用全部数据

使用最佳训练策略

## 训练次级学习器

模型融合：基础学习器

## 模型融合：次级学习器

采用加权平均，权重取二级模型评分。

* 注意 sr\_rss\_help\_bonus\_level > 0 的玩家是付费的。
* 训练集的玩家注册时间长118，测试集的时间短89，预测值按比例118/89看看。