

乐信机器学习大赛

蒋老师的学生



目录

[1.项目背景概括 3](#_Toc493858122)

[2.项目思路 & 项目流程 4](#_Toc493858123)

[3.数据探索分析 5](#_Toc493858124)

[4.数据处理 6](#_Toc493858125)

[1)不平衡样本处理方法 6](#_Toc493858126)

[2)缺失值处理 6](#_Toc493858127)

[3)文本型变量处理 6](#_Toc493858128)

[5.特征工程 7](#_Toc493858129)

[预测用户未来6个月是否逾期特征分析过程 7](#_Toc493858130)

[6.模型构建 12](#_Toc493858131)

[1）方法论概述 12](#_Toc493858132)

[2）模型比较与选择 12](#_Toc493858133)

[3）模型细节 12](#_Toc493858134)

[4）模型训练结果 12](#_Toc493858135)

[7.模型测试 13](#_Toc493858136)

[8.模型局限性讨论 13](#_Toc493858137)

[9.系统实施方案讨论 13](#_Toc493858138)

[技术方案 13](#_Toc493858139)

[商业逻辑方案 14](#_Toc493858140)

[10. 团队介绍 14](#_Toc493858141)

[11. APPENDIX － 竞赛详细代码 15](#_Toc493858142)

## 项目背景概括

根据IBM的研究表明，全球每年欺诈令金融行业损失大约800亿美元。在美国，每年信用卡和借记卡发行商的损失就有24亿美元。为了减少损失，各大公司使用了各种方法建立用户的信用指数模型，来估计用户的信用度。近些年，随着机器学习的兴起，借助机器学习的帮助，可以更高效实现欺诈检测技术。

此次，乐信公司举办LexinFintech技术大赛，旨在鼓励使用机器学习方法预判用户未来的风险，评估用户的消费需求，并通过预测结果有效地制定额度政策，投放资源给优质用户，提高整体资金的使用效率。

## 项目思路 & 项目流程

（注：auc特征提取表示为估计用户逾期概率进行的特征提取，mae特征提取表示为估计用户消费金额进行的特征提取）

数据探索

数据合并

预测

模型融合

XGBOOST模型探索特征权重

auc特征提取

mae特征提取

## 数据探索分析

训练集和测试集的三个文件p6m、p12m、loginscene分别表示用户过去六个月订单行为汇总信息，过去12个月用户的订单金额信息，过去6个月用户场景行为信息。以过去六个月订单行为汇总信息举例说明我们的分析过程，文件中共有300000条记录，每条记录有130个字段。其中记录了50000个用户过去六个月的订单行为信息，每条记录对应一个用户的一个月份的订单信息。根据模型训练要求，需将每个用户的6条记录合并为一条记录，对记录中的每个字段添加对应的月份作为后缀（如c\_log\_eqp\_dist\_cntJUL16表示2016年7月份用户登录时使用的不同设备数的字段信息），并删除当前观测月还款日和观测月两个字段，得到一个新的文件。新文件中有50000条记录，每条记录对应一个用户，每条记录763个字段信息。按照同样的方式处理训练集和测试集的文件p12m和loginscene，在此不再赘述。

观察发现训练集和测试集中的p6m文件的初始观察月相差两个月份，训练集的p6m文件记录了用户从2016年5月至2016年10月的订单行为信息，测试机的p6m文件记录了用户从2016年7月至2016年12月的订单行为信息，由于两个文件均是记录用户过去六个月的订单信息，且是预测用户未来六个月的消费情况，为保证字段名一致，可将测试集p6m文件的字段名的后缀月份提前两个月（如将c\_log\_eqp\_dist\_cntSEP16改为c\_log\_eqp\_dist\_cntJUL16）。

## 数据处理

### 不平衡样本处理方法

训练集中负样本数目（逾期用户）为1500条，正样本数目（未逾期用户）为48500条；为避免过拟合，将负样本复制3份加入到训练集中，此时训练集中负样本数目为6000条，正样本数目为48500条。

### 缺失值处理

在特征提取中，为符合常理，将缺失值nan填充为0，异常值inf或-inf填充为0。如当月注册场景次数（c\_scene\_reg\_tot\_cnt）为0，月浏览场景总次数为0，为提取用户的月注册场景次数比例特征，用月注册场景次数除以月浏览场景总次数计算月注册场景次数比例时，会产生inf值；此时将inf值填充为0，比较符合常理。同理，在提取特征的过程中，若存在缺失值nan，填充为0。

### 文本型变量处理

分析数据，总共存在三种类型：数值型，string型和timestamp型；数据中的string型对应的是日期和MD5加密后的用户ID，timestamp型都对应的是日期，由于日期不适合作为特征属性，而是对应日期下的行为数据作为特征属性，故将string型和timestamp型对应的日期全部删除，而用户ID需要作为每条记录的标识号保留，只剩下数值型的特征。

## 特征工程

对于预测用户未来6个月是否逾期以及未来6个月的消费金额两个问题，我们进行了不同的特征筛选与处理过程。

### 预测用户未来6个月是否逾期特征分析过程

规则一：分析当用户过去6个月每个月的待还总服务费（如dh\_tot\_feeJUL16）以及待还总服务费所占应还总服务费的比例（如dh\_tot\_feePercentJUL16），若每个月的待还总服务费越高，可以预测未来6个月用户的待还总服务费会比较高，有可能会出现逾期的情况；

规则二：分析当用户过去6个月每个月的待还月服务费（如dh\_mon\_feePercentJUL16）以及待还月服务费所占应还月服务费的比例（如dh\_mon\_feePercentJUL16），若每个月的待还月服务费越高，可以预测未来6个月用户的待还月服务费会比较高，有可能会出现逾期的情况；

规则三：分析当用户过去6个月每个月的应还本金（如paying\_capitalJUL16）以及待还本金所占应还本金的比例（如dh\_balPercentJUL16），若每个月的待还本金越高，可以预测未来6个月用户的待还本金会比较高，有可能会出现逾期的情况；

规则四：分析当用户过去6个月每个月的月待还款额（如dh\_cpt\_pymtJUL16）以及月待还款额所占应还款额的比例（如dh\_cpt\_pymtPercentJUL16），若每个月的月待还款额越高，可以预测未来6个月用户的月待还款额会比较高，有可能会出现逾期的情况；

规则五：分析用户过去6个月的新建分期类订单金额，若每个新建的分期类订单金额比例较高，可以以此预测未来6个月用户新建分期类的订单金额较高，更可能出现逾期的情况，故计算各个月新建分期类订单金额比例（如od\_zdfq\_brwPercentJUL16）；

规则六：我们采用xgboost对训练数据进行了初步的预估，发现各个月已还分期类服务费特征权重较高，故计算各个月已还分期类服务费所占比例（如payed\_zdfq\_mon\_feePercentJUL16）。

同时，计算上述六条规则中待还总服务费，待还总服务费比例，待还月服务费，待还月服务费比例，待还本金，待还本金比例，月待还款额，月待还款额比例，新建分期类订单金额比例，已还分期类服务费比例的平均值。

规则七：观察用户过去每个月应还完的订单数以及过去每个用户已还完订单数的比例，若用户过去6个月应还完的订单数越少，可以预测未来6个月用户应还完的订单数比较少，更可能发生逾期的情况。故计算用户月已还完订单数的比例。

规则八：计算用户过去每个月提前还完的订单比例，若提前还完的订单比例越高，则可以预测用户未来6个月逾期的概率较低。

规则九：计算用户过去每个月已还完的本金金额比例，若已还完的本金金额比例越高，则可以预测用户未来6个月逾期的概率较低。

规则十：计算用户过去每个月提前还完的本金金额比例，若提前还完的本金金额比例越高，则可以预测用户未来6个月逾期的概率较低。

规则十一：计算用户过去6个月总的逾期天数所占首个观察月历史逾期天数的比例，若过去6个月总的逾期天数所占历史逾期天数的比例越高，则可以预测用户未来6个月逾期的概率较高。

规则十一：计算用户过去6个月总的逾期账期数所占首个观察月历史逾期账期数的比例，若过去6个月总的逾期天数所占历史逾期天数的比例越高，则可以预测用户未来6个月逾期的概率较高。

规则十二：观察用户过去6个月的月额度是否提高，如最后一个月的月额度较首月的月额度有提高，可以认为过去6个月的消费行为良好，未来6个月逾期的概率较低。

规则十三：分析用户过去六个月的新增订单明细数据，根据用户每个月还款日，计算还款日之前20天的账单日；分析用户每个月账单日之前的新建分期数以及新建订单金额，若分期数越多，订单金额越高，则未来6个月需要还款额越高，更有可能导致用户未来6个月逾期。

为观察每个月截止到当前应还款日的已还本金（分）的变化率，求过去6个月截止到当前应还款日已还本金的方差，得到新特征payed\_capitalfeavar；

1. 为观察每个月截止到当前应还款日的已还实际现金本金（分）的变化率，求过去6个月的已还实际现金本金（分）的方差，得到新特征payed\_actual\_capitalfeavar；
2. 为观察每个月截止到当前应还款日的已还虚拟现金本金（分）的变化率，求过去6个月截止到当前应还款日已还虚拟现金本金的方差，得到新特征payed\_virtual\_capitalfeavar；
3. 为观察每个月剩余可用额度（分）的变化率，求过去6个月的剩余可用额度（分）的方差，得到新特征fopen\_to\_buyfeavar；
4. 为观察每个月截止到当前应还款日的应还账单延期类总服务费（分）的变化率，求过去6个月截止到当前应还款日的应还账单延期类总服务费（分）的方差，得到新特征zdyq\_paying\_tot\_feefeavar；
5. 为观察每个月历史存量创建现金类订单金额（分）(减首付后)的变化率，求过去6个月的历史存量创建现金类订单金额（分）(减首付后)的方差，得到新特征payed\_actual\_capitalfeavar；
6. 为观察每个月截止到当前应还款日的已还虚拟现金本金（分）的变化率，求过去6个月截止到当前应还款日已还虚拟现金本金的方差，得到新特征payed\_virtual\_capitalfeavar；
7. 为观察每个月剩余可用额度（分）的变化率，求过去6个月的剩余可用额度（分）的方差，得到新特征fopen\_to\_buyfeavar；
8. 计算各个月通过PC浏览场景的次数比例，如c\_scene\_pc\_tot\_cntPercentJUL16；
9. 计算各个月通过APP浏览场景的次数比例，如c\_scene\_app\_tot\_cntPercentJUL16；
10. 计算各个月通过H5浏览场景的次数比例，如c\_scene\_h5\_tot\_cntPercentJUL16；
11. 计算各个月通过Android浏览场景的次数比例，如c\_scene\_android\_tot\_cntPercentJUN16；
12. 计算各个月通过IOS浏览场景的次数比例，如c\_scene\_ios\_tot\_cntPercentJUL16；
13. 分别计算过去6个月通过PC浏览场景的平均次数，通过APP浏览场景的平均次数，通过Android浏览场景的平均次数和通过IOS浏览场景的平均次数，得到新特征avgc\_scene\_pc\_tot\_cnt, avgc\_scene\_app\_tot\_cnt, avgc\_scene\_h5\_tot\_cnt, avgc\_scene\_android\_tot\_cnt, avgc\_scene\_ios\_tot\_cnt;
14. 计算各个月场景为注册的次数比例，如c\_scene\_reg\_tot\_cntPercentJUL16
15. 计算各个月场景为登录的次数比例：c\_scene\_dl\_tot\_cntPercentJUL16
16. 计算各个月场景为下单的次数比例：c\_scene\_od\_tot\_cntPercentJUL16
17. 计算各个月场景为还款的次数比例：c\_scene\_rp\_tot\_cntPercentJUL16
18. 计算各个月场景为修改信息的次数比例：c\_scene\_xgxx\_tot\_cntPercentJUL16
19. 计算各个月场景为评论商品的次数比例：c\_scene\_plsp\_tot\_cntPercentJUL16
20. 各个月场景为上传资料的次数比例：c\_scene\_sczl\_tot\_cntPercentJUL16
21. 各个月场景为商户相关的次数比例：c\_scene\_sh\_tot\_cntPercentJUL16

### 预测用户未来6个月消费金额特征工程

预测用户未来6个月消费金额数据处理：首先我们对清洗并且合并之后的数据通过XGBOOST进行初步的验证，并且打印出特征的重要程度。

数据分析：我们根据打印出来的特征的重要程度对数据进行分析。

（1）我们发现用户六个月订单行为汇总当中的特征，用户最近的消费金额等特征对未来6个月消费总额的影响更大，所以我们根据模型计算出来的特征的重要度乘上用户当前月的消费金额来构造出新的特征。其中包括了（用户当前观测月新建订单金额、当前观测月新建非延期分期金额、当前观测月新建延期分期订单金额、当前观测月新建3c类订单金额等）

（2）计算过去六个月订单行为汇总表当中，用户过去6个月相关特征（用户当前观测月新建订单金额、当前观测月新建非延期分期金额、当前观测月新建延期分期订单金额、当前观测月新建3c类订单金额、历史存量创建订单金额）等特征的平均值。

## 模型构建

### 未来6个月逾期概率的预测

对于用户未来6个月逾期概率，我们采用了XGBClassifier\_A+XGBClassifier\_B+RandomForestClassifier三个模型进行stacking，选取了XGBClassifier\_C作为stacker

具体stacking过程如下：

1首先对训练数据分为5折，然后对每一模型进行训练，并且每次训练过程当中对测试集进行预测和对训练数据剩余部分进行预测，当一个模型训练完成之后计算该模型的5次预测的平均值并加入到数组当中，这样重复直到所有模型都训练完毕。

2使用预测出的训练集的概率作为staker的训练集，以预测出的预测集的概率作为stacker新的预测集。

### 未来6个月消费金额的预测

对于未来6个月消费金额我们选取了XGBRegressor+RandomForestRegressor进行stacking，选取了Ridge()作为stacker

（1）首先对训练数据分为5折，然后对每一模型进行训练，并且每次训练过程当中对测试集进行预测和对训练数据剩余部分进行预测，当一个模型训练完成之后计算该模型的5次预测的平均值并加入到数组当中，这样重复直到所有模型都训练完毕。

（2）使用预测出的训练集的概率作为stacker的训练集，以预测出的预测集的概率作为stacker新的预测集。

### 模型细节

Ubuntu 16.04 python2.7

## 模型测试

为了判断模型的泛化能力，我们在训练集上对每个模型均采用五折交叉验证，并使用网格搜索调整模型参数，根据测试结果选择每个模型参数。

## 模型局限性讨论

无法解决冷启动的问题，无法预测新用户的未来6个月的逾期概率以及消费金额

## 系统实施方案讨论

### 技术方案

为了在真实商业场景下使用，先收集用户的历史数据，包括用户个人信息，用户历史订单行为汇总信息，用户历史订单金额信息，用户历史场景行为信息，以6个月为周期划分历史数据，如划分得到N个阶段的历史数据，并编号为1~N；用编号1的数据作为训练集，编号为2的数据作为测试集，以此类推，用编号为N-1的数据作为训练集，编号为N的数据为测试集。然后建立模型，用训练集训练，用测试集评估模型预测能力，不断调整参数。

为解决冷启动的问题，若要预估新用户是否逾期以及消费金额，可采用协同过滤的方法来初步预测。

### 商业逻辑方案

模型建立后，根据模型的预测结果，若用户逾期概率超过0.5，则认为用户未来6个月很可能会逾期，可降低用户的月额度，并要求用户每次新建订单时，需要支付较高比例的首付金额。

## 团队介绍

队员曹进，北京航空航天大学计算机学院研究生二年级在读

队员吴志新，北京航空航天大学软件学院研究生二年级在读

## APPENDIX － 竞赛详细代码

#coding=utf-8

###数据处理

import pandas as pd

import numpy as np

import csv

from testFileConfig import \*

from trainFileConfig import \*

import json

import xlrd

class utils:

@staticmethod

def rowTransform(dataFrame,cyc\_date,rowIndex):

index = dataFrame.columns

length=dataFrame.shape[0]

cols=dataFrame.shape[1]

all\_index = []

data\_Dict = {}

data\_Dict['fuid\_md5'] = dataFrame.iloc[0][0]

all\_index.append('fuid\_md5')

for val in index[1:]:

for i in xrange(0,length):

date = "".join(cyc\_date[i].split('-')[0:2])

tmpval = val + date

all\_index.append(tmpval)

# print all\_index.\_\_len\_\_()

ls\_index=1

for j in xrange(1,cols):

for i in xrange(0,length):

data\_Dict[all\_index[ls\_index]] = dataFrame.iloc[i][j]

ls\_index+=1

return pd.DataFrame(data\_Dict,columns=all\_index,index=[rowIndex])

"""记录合并"""

@staticmethod

def transformDataFrame(file,outfile):

tmpSet = set()

tmpSet.add('fuid\_md5')

dataDict = {}

csv\_reader = csv.reader(open(file))

flag = 0

# count = 0

for row in csv\_reader:

# count+=1

# if count==26:

# break

if flag == 0:

columns = row

flag+=1

else:

if row[0] not in dataDict:

dataDict[row[0]] = {}

tmpKey = row[1][2:7]

for i in range(3,columns.\_\_len\_\_()):

dataDict[row[0]][columns[i]+tmpKey]=row[i]

tmpSet.add(columns[i]+tmpKey)

# print dataDict

resDict = {}

columnsName = list(tmpSet)

# print columnsName

# columnsName.extend(dataDict['1f9aa4769d4cee1656a17c5546b95839'].keys())

for val in columnsName:

resDict[val]=[]

for keyStr in dataDict:

resDict['fuid\_md5'].append(keyStr)

for tmpKey in dataDict[keyStr]:

# if tmpKey not in resDict:

# resDict[tmpKey]=[]

resDict[tmpKey].append(dataDict[keyStr][tmpKey])

# print resDict

# for tmpDict in dataDict:

# for keyStr in tmpDict:

# resDict[keyStr].append(tmpDict[keyStr])

# allColumns = resDict.keys()

# for val in columns:

# for tmpVal in allColumns:

# if tmpVal[0:-6] == val:

# columnsName.append(tmpVal)

print resDict.\_\_len\_\_()

# dataFrame = pd.DataFrame(resDict,columns=columnsName)

# dataFrame.to\_csv(outfile,index=False)

@staticmethod

def reName6Month(file,outfile):

p6\_DF = pd.read\_csv(file)

columnsName = p6\_DF.columns

res\_columns = []

tmpLs = []

for val in columnsName:

tmpLs.append(val)

if val == 'fuid\_md5':

res\_columns.append(val)

else:

tmpStr = val[-5:]

if tmpStr == 'JUL16':

reStr = 'MAY16'

elif tmpStr == 'AUG16':

reStr = 'JUN16'

elif tmpStr == 'SEP16':

reStr = 'JUL16'

elif tmpStr == 'OCT16':

reStr = 'AUG16'

elif tmpStr == 'NOV16':

reStr = 'SEP16'

else:

reStr = 'OCT16'

restr = val[0:-5] + reStr

res\_columns.append(restr)

# print tmpLs

# print res\_columns

# p6\_DF.rename(res\_columns,inplace=True)

p6\_DF.columns = res\_columns

p6\_DF.to\_csv(outfile,index=False)

"""将测试集合中的时间减少两个月份"""

@staticmethod

def reName12Month(file, outfile):

p12\_DF = pd.read\_csv(file)

columnsName = p12\_DF.columns

res\_columns = []

tmpLs = []

for val in columnsName:

tmpLs.append(val)

if val == 'fuid\_md5':

res\_columns.append(val)

else:

tmpStr = val[-5:]

if tmpStr == 'JAN16':

reStr = 'NOV15'

elif tmpStr == 'FEB16':

reStr = 'DEC15'

elif tmpStr == 'MAR16':

reStr = 'JAN16'

elif tmpStr == 'APR16':

reStr = 'FEB16'

elif tmpStr == 'MAY16':

reStr = 'MAR16'

elif tmpStr == 'JUN16':

reStr = 'APR16'

elif tmpStr == 'JUL16':

reStr = 'MAY16'

elif tmpStr == 'AUG16':

reStr = 'JUN16'

elif tmpStr == 'SEP16':

reStr = 'JUL16'

elif tmpStr == 'OCT16':

reStr = 'AUG16'

elif tmpStr == 'NOV16':

reStr = 'SEP16'

elif tmpStr == 'DEC16':

reStr = 'OCT16'

restr = val[0:-5] + reStr

res\_columns.append(restr)

# print tmpLs

# print res\_columns

# p6\_DF.rename(res\_columns,inplace=True)

p12\_DF.columns = res\_columns

p12\_DF.to\_csv(outfile, index=False)

###特征提取

特征处理

#coding=utf-8

import pandas as pd

import numpy as np

p6m\_DF1 = pd.read\_csv('../train\_file/特征处理后的p6M\_mdl.csv')

p6m\_DF2 = pd.read\_csv('../test\_file/经过特征处理后的p6M\_offtime1.csv')

p6m\_DF = pd.concat([p6m\_DF1,p6m\_DF2],axis=0)

# print p6m\_DF1.shape

# print p6m\_DF2.shape

print p6m\_DF.shape

result = p6m\_DF[['fuid\_md5']]

featureLs = ['od\_cnt','actual\_od\_cnt','virtual\_od\_cnt','od\_zdfq\_cnt','od\_brw',

'actual\_od\_brw','virtual\_od\_brw','od\_zdfq\_brw','cumu\_od\_cnt','cumu\_actual\_od\_cnt'

,'cumu\_virtual\_od\_cnt','cumu\_od\_zdfq\_cnt','cumu\_od\_brw','cumu\_actual\_od\_brw',

'cumu\_virtual\_od\_brw','cumu\_od\_zdfq\_brw','payed\_capital','payed\_actual\_capital',

'payed\_virtual\_capital','payed\_zdfq\_capital','payed\_mon\_fee','payed\_zdfq\_mon\_fee',

'payed\_tot\_fee','payed\_zdfq\_tot\_fee','bal','zdfq\_bal','paying\_mon\_fee','zdfq\_paying\_mon\_fee'

,'paying\_tot\_fee','zdfq\_paying\_tot\_fee','paying\_complete\_od\_cnt','payed\_complete\_od\_cnt',

'paying\_complete\_od\_brw','payed\_complete\_od\_brw',

'acre\_repay\_od\_cnt','acre\_repay\_od\_cpt','foverdue\_paying\_day',

'foverdue\_paying\_cyc','foverdue\_payed\_day','foverdue\_payed\_cyc','cpt\_pymt','credit\_limit',

'fcredit\_update\_time','futilization','fopen\_to\_buy']

for mon in ['JUL16','OCT16','AUG16','SEP16','JUN16','MAY16']:

for fea in featureLs:

result[fea+mon] = p6m\_DF[fea+mon]

for mon in ['JUL16','OCT16','AUG16','SEP16','JUN16','MAY16']:

##应还总服务费-已还总服务费

result['dh\_tot\_fee' + mon] = p6m\_DF['paying\_tot\_fee' + mon] - p6m\_DF['payed\_tot\_fee' + mon]

##待还总服务费所占比率

result['dh\_tot\_feePercent' + mon] = result['dh\_tot\_fee' + mon].divide(p6m\_DF['paying\_tot\_fee' + mon],fill\_value = 0)

##应还月服务费-已还月服务费

result['dh\_mon\_fee' + mon] = p6m\_DF['paying\_mon\_fee' + mon] - p6m\_DF['payed\_mon\_fee' + mon]

##待还月服务费所占比率

result['dh\_mon\_feePercent' + mon] = result['dh\_mon\_fee' + mon].divide(p6m\_DF['paying\_mon\_fee' + mon],fill\_value = 0)

##已还本金+待还本金

result['paying\_capital' + mon] = p6m\_DF['payed\_capital' + mon] + p6m\_DF['bal' + mon]

##待还本金比率

result['dh\_balPercent' + mon] = p6m\_DF['bal' + mon].divide(result['paying\_capital' + mon],fill\_value = 0)

"""月待还款额 = 月额度 - 已还款额 - 剩余可用额度"""

result['dh\_cpt\_pymt' + mon] = p6m\_DF['credit\_limit' + mon] - p6m\_DF['cpt\_pymt' + mon] - p6m\_DF['fopen\_to\_buy' + mon]

##月待还款比例

result['dh\_cpt\_pymtPercent' + mon] = result['dh\_cpt\_pymt' + mon].divide( (p6m\_DF['credit\_limit' + mon] - p6m\_DF['fopen\_to\_buy' + mon]),fill\_value = 0)

"""月新建分期类比例 od\_zdfq\_brw"""

result['od\_zdfq\_brwPercent'+mon] = p6m\_DF['od\_zdfq\_brw'+mon] / p6m\_DF['od\_brw'+mon]

"""已还分期类月服务费所占比率"""

result['payed\_zdfq\_mon\_feePercent'+mon] = p6m\_DF['payed\_zdfq\_mon\_fee'+mon] / p6m\_DF['payed\_mon\_fee'+mon]

for feature in ['dh\_tot\_fee','dh\_tot\_feePercent','dh\_mon\_fee','dh\_mon\_feePercent',

'paying\_capital','dh\_balPercent','dh\_cpt\_pymt','dh\_cpt\_pymtPercent']:

result['avg' + feature] = (result[feature + 'JUL16'] + result[feature + 'OCT16'] + result[feature + 'AUG16'] + \

result[feature + 'SEP16'] + result[feature + 'JUN16'] + result[feature + 'MAY16']) / 6.0

##计算用户MAY16（2016年5月），JUN16，JUL16，AUG16，SEP16应还完的订单数

result['paying\_complete\_od\_cnt\_InMAY16'] = p6m\_DF['paying\_complete\_od\_cntJUN16'] - p6m\_DF['paying\_complete\_od\_cntMAY16']

result['paying\_complete\_od\_cnt\_InJUN16'] = p6m\_DF['paying\_complete\_od\_cntJUL16'] - p6m\_DF['paying\_complete\_od\_cntJUN16']

result['paying\_complete\_od\_cnt\_InJUL16'] = p6m\_DF['paying\_complete\_od\_cntAUG16'] - p6m\_DF['paying\_complete\_od\_cntJUL16']

result['paying\_complete\_od\_cnt\_InAUG16'] = p6m\_DF['paying\_complete\_od\_cntSEP16'] - p6m\_DF['paying\_complete\_od\_cntAUG16']

result['paying\_complete\_od\_cnt\_InSEP16'] = p6m\_DF['paying\_complete\_od\_cntOCT16'] - p6m\_DF['paying\_complete\_od\_cntSEP16']

##计算用户MAY16，JUN16，JUL16，AUG16，SEP16已还完的订单比率

result['payed\_complete\_od\_cntMAY16Percent'] = p6m\_DF['payed\_complete\_od\_cntMAY16'].divide(result['paying\_complete\_od\_cnt\_InMAY16'],fill\_value = 0)

result['payed\_complete\_od\_cntJUN16Percent'] = p6m\_DF['payed\_complete\_od\_cntJUN16'].divide(result['paying\_complete\_od\_cnt\_InJUN16'],fill\_value = 0)

result['payed\_complete\_od\_cntJUL16Percent'] = p6m\_DF['payed\_complete\_od\_cntJUL16'].divide(result['paying\_complete\_od\_cnt\_InJUL16'],fill\_value = 0)

result['payed\_complete\_od\_cntAUG16Percent'] = p6m\_DF['payed\_complete\_od\_cntAUG16'].divide(result['paying\_complete\_od\_cnt\_InAUG16'],fill\_value = 0)

result['payed\_complete\_od\_cntSEP16Percent'] = p6m\_DF['payed\_complete\_od\_cntSEP16'].divide(result['paying\_complete\_od\_cnt\_InSEP16'],fill\_value = 0)

##计算用户MAY16，JUN16，JUL16，AUG16，SEP16提前还完的订单数

result['acre\_repay\_od\_cntInMAY16'] = p6m\_DF['acre\_repay\_od\_cntJUN16'] - p6m\_DF['acre\_repay\_od\_cntMAY16']

result['acre\_repay\_od\_cntInJUN16'] = p6m\_DF['acre\_repay\_od\_cntJUL16'] - p6m\_DF['acre\_repay\_od\_cntJUN16']

result['acre\_repay\_od\_cntInJUL16'] = p6m\_DF['acre\_repay\_od\_cntAUG16'] - p6m\_DF['acre\_repay\_od\_cntJUL16']

result['acre\_repay\_od\_cntInAUG16'] = p6m\_DF['acre\_repay\_od\_cntSEP16'] - p6m\_DF['acre\_repay\_od\_cntAUG16']

result['acre\_repay\_od\_cntInSEP16'] = p6m\_DF['acre\_repay\_od\_cntOCT16'] - p6m\_DF['acre\_repay\_od\_cntSEP16']

##计算用户MAY16，JUN16，JUL16，AUG16，SEP16提前还完的订单比率

result['acre\_repay\_od\_cntInMAY16Percent'] = result['acre\_repay\_od\_cntInMAY16'].divide(result['paying\_complete\_od\_cnt\_InMAY16'],fill\_value = 0)

result['acre\_repay\_od\_cntInJUN16Percent'] = result['acre\_repay\_od\_cntInJUN16'].divide(result['paying\_complete\_od\_cnt\_InJUN16'],fill\_value = 0)

result['acre\_repay\_od\_cntInJUL16Percent'] = result['acre\_repay\_od\_cntInJUL16'].divide(result['paying\_complete\_od\_cnt\_InJUL16'],fill\_value = 0)

result['acre\_repay\_od\_cntInAUG16Percent'] = result['acre\_repay\_od\_cntInAUG16'].divide(result['paying\_complete\_od\_cnt\_InAUG16'],fill\_value = 0)

result['acre\_repay\_od\_cntInSEP16Percent'] = result['acre\_repay\_od\_cntInSEP16'].divide(result['paying\_complete\_od\_cnt\_InSEP16'],fill\_value = 0)

##计算用户MAY16，JUN16，JUL16，AUG16，SEP16应还完的本金金额

result['paying\_complete\_od\_brw\_InMAY16'] = p6m\_DF['paying\_complete\_od\_brwJUN16'] - p6m\_DF['paying\_complete\_od\_brwMAY16']

result['paying\_complete\_od\_brw\_InJUN16'] = p6m\_DF['paying\_complete\_od\_brwJUL16'] - p6m\_DF['paying\_complete\_od\_brwJUN16']

result['paying\_complete\_od\_brw\_InJUL16'] = p6m\_DF['paying\_complete\_od\_brwAUG16'] - p6m\_DF['paying\_complete\_od\_brwJUL16']

result['paying\_complete\_od\_brw\_InAUG16'] = p6m\_DF['paying\_complete\_od\_brwSEP16'] - p6m\_DF['paying\_complete\_od\_brwAUG16']

result['paying\_complete\_od\_brw\_InSEP16'] = p6m\_DF['paying\_complete\_od\_brwOCT16'] - p6m\_DF['paying\_complete\_od\_brwSEP16']

##计算用户MAY16，JUN16，JUL16，AUG16，SEP16已还完的本金金额比率

result['payed\_complete\_od\_brwMAY16Percent'] = p6m\_DF['paying\_complete\_od\_brwMAY16'].divide(result['paying\_complete\_od\_brw\_InMAY16'],fill\_value = 0)

result['payed\_complete\_od\_brwJUN16Percent'] = p6m\_DF['paying\_complete\_od\_brwJUN16'].divide(result['paying\_complete\_od\_brw\_InJUN16'],fill\_value = 0)

result['payed\_complete\_od\_brwJUL16Percent'] = p6m\_DF['paying\_complete\_od\_brwJUL16'].divide(result['paying\_complete\_od\_brw\_InJUL16'],fill\_value = 0)

result['payed\_complete\_od\_brwAUG16Percent'] = p6m\_DF['paying\_complete\_od\_brwAUG16'].divide(result['paying\_complete\_od\_brw\_InAUG16'],fill\_value = 0)

result['payed\_complete\_od\_brwSEP16Percent'] = p6m\_DF['paying\_complete\_od\_brwSEP16'].divide(result['paying\_complete\_od\_brw\_InSEP16'],fill\_value = 0)

##计算用户MAY16，JUN16，JUL16，AUG16，SEP16提前还款的本金金额

result['acre\_repay\_od\_cptInMAY16'] = p6m\_DF['acre\_repay\_od\_cptJUN16'] - p6m\_DF['acre\_repay\_od\_cptMAY16']

result['acre\_repay\_od\_cptInJUN16'] = p6m\_DF['acre\_repay\_od\_cptJUL16'] - p6m\_DF['acre\_repay\_od\_cptJUN16']

result['acre\_repay\_od\_cptInJUL16'] = p6m\_DF['acre\_repay\_od\_cptAUG16'] - p6m\_DF['acre\_repay\_od\_cptJUL16']

result['acre\_repay\_od\_cptInAUG16'] = p6m\_DF['acre\_repay\_od\_cptSEP16'] - p6m\_DF['acre\_repay\_od\_cptAUG16']

result['acre\_repay\_od\_cptInSEP16'] = p6m\_DF['acre\_repay\_od\_cptOCT16'] - p6m\_DF['acre\_repay\_od\_cptSEP16']

##计算用户MAY16，JUN16，JUL16，AUG16，SEP16提前还款的本金金额比率

result['acre\_repay\_od\_cptInMAY16Percent'] = result['acre\_repay\_od\_cptInMAY16'].divide(result['paying\_complete\_od\_brw\_InMAY16'],fill\_value = 0)

result['acre\_repay\_od\_cptInJUN16Percent'] = result['acre\_repay\_od\_cptInJUN16'].divide(result['paying\_complete\_od\_brw\_InJUN16'],fill\_value = 0)

result['acre\_repay\_od\_cptInJUL16Percent'] = result['acre\_repay\_od\_cptInJUL16'].divide(result['paying\_complete\_od\_brw\_InJUL16'],fill\_value = 0)

result['acre\_repay\_od\_cptInAUG16Percent'] = result['acre\_repay\_od\_cptInAUG16'].divide(result['paying\_complete\_od\_brw\_InAUG16'],fill\_value = 0)

result['acre\_repay\_od\_cptInSEP16Percent'] = result['acre\_repay\_od\_cptInSEP16'].divide(result['paying\_complete\_od\_brw\_InSEP16'],fill\_value = 0)

##6个月逾期的比率

result['allfoverdue\_paying\_day'] = p6m\_DF['foverdue\_paying\_dayMAY16']

result['allfoverdue\_paying\_cyc'] = p6m\_DF['foverdue\_paying\_cycMAY16']

for mon in ['JUL16','OCT16','AUG16','SEP16','JUN16']:

result['allfoverdue\_paying\_day'] = result['allfoverdue\_paying\_day'] + p6m\_DF['foverdue\_paying\_day' + mon]

result['allfoverdue\_paying\_cyc'] = result['allfoverdue\_paying\_cyc'] + p6m\_DF['foverdue\_paying\_cyc' + mon]

result['foverdue\_paying\_dayPercent'] = result['allfoverdue\_paying\_day'].divide(p6m\_DF['foverdue\_payed\_dayMAY16'],fill\_value = 0)

##过去6个月的逾期天数 / 5月的用户历史逾期天数

result['foverdue\_paying\_cycPercent'] = result['allfoverdue\_paying\_cyc'].divide(p6m\_DF['foverdue\_payed\_cycMAY16'],fill\_value = 0)

##额度是否增加

result['credit\_limit'] = p6m\_DF['credit\_limitOCT16'] - p6m\_DF['credit\_limitMAY16']

result = result.replace(-np.inf,0)

result = result.replace(np.inf,0)

result = result.fillna(0)

print result.shape

result.iloc[:50000,:].to\_csv('../train\_file/p6m挖掘出的新特征(删除部分特征).csv', index=False)

result.iloc[50000:,:].to\_csv('../test\_file/p6m挖掘出的新特征(删除部分特征).csv', index=False)