融360
RONG360.COM

Prediction of the second loan on Rong 360

SHU LIU

1. Introduction

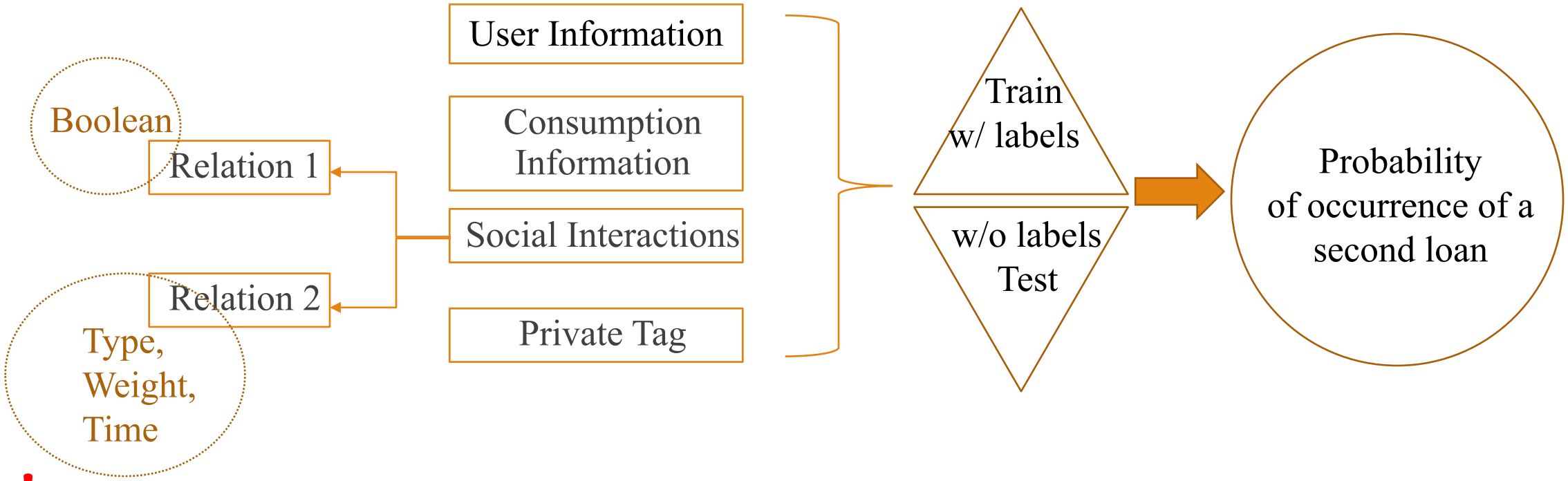
is a financial products search and recommendation engine that enables users to compare loan products from various banks and lending agencies.
In a word, Rong 360 is a platform that matches institutions who have money with people who need to borrow money.

Funding Rounds (5) - $258M

Date	Amount / Round	Valuation	Lead Investor	Investors
Oct, 2015	$158M / Series D	–	Sailing Capital YF Capital (Yunfeng Capital)	4
Jul, 2014	$60M / Series C	–	Pavillion Capital	3
Jul, 2013	$30M / Series B	–	Sequoia Capital	2
May, 2012	$10M / Series A	–	Lightspeed Venture Partners	4
Jan, 2012	undisclosed amount / Seed	–	–	2

2. Dataset

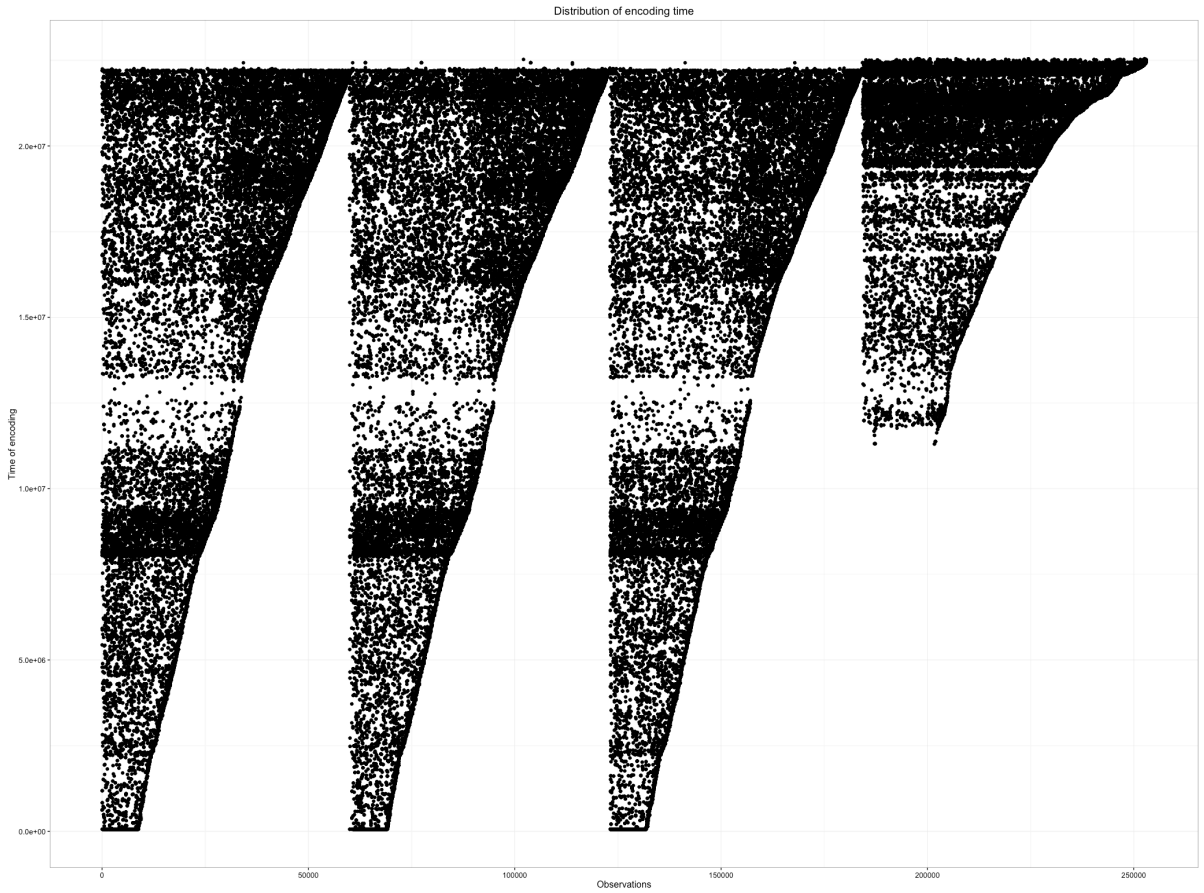
I checked basic structures of all datasets, including information duplication, missingness, and intersection of observations among datasets.



- ! Challenge I: A same problem in above four datasets: Multiple rows with repeated user_id
- ! Challenge II: Disordered missingness
- ! Challenge III: Incomplete inforamtion of user_ids from different datasets

3. EDA

3.1 User_id Duplication:
I sort observations by index, and check numeric distribution of the variable ‘tm_encode’



If I sort observations by index,
and graph it with the variable ’tm_encod’,
it’s obvious that an ‘uneven right-angled triangle’ reappeared for about four times.

This graph implies a new categorical variable for feature engineering

3. EDA

3.1 User_id Duplication:

I found duplication of user_id comes from two reasons.

1. Add new information

	user_id	age	sex	expect_quota	max_month_repay	occupation	education	marital_status	liv
7	ec76ffd36a9990016ad1e9a4888c3291	NONE	0	50000	*NA*	2	2	3	
8	ec76ffd36a9990016ad1e9a4888c3291	41	1	50000	*NA*	2	2	3	

2. Revise old information

	user_id	age	sex	expect_quota	max_month_repay	occupation	education	marital_status	liv
179	7c6c781dfe97819f580c097f7dc4791e	NONE	0	2000	*NA*	2	3	1	
180	7c6c781dfe97819f580c097f7dc4791e	NONE	0	5000	*NA*	2	3	1	

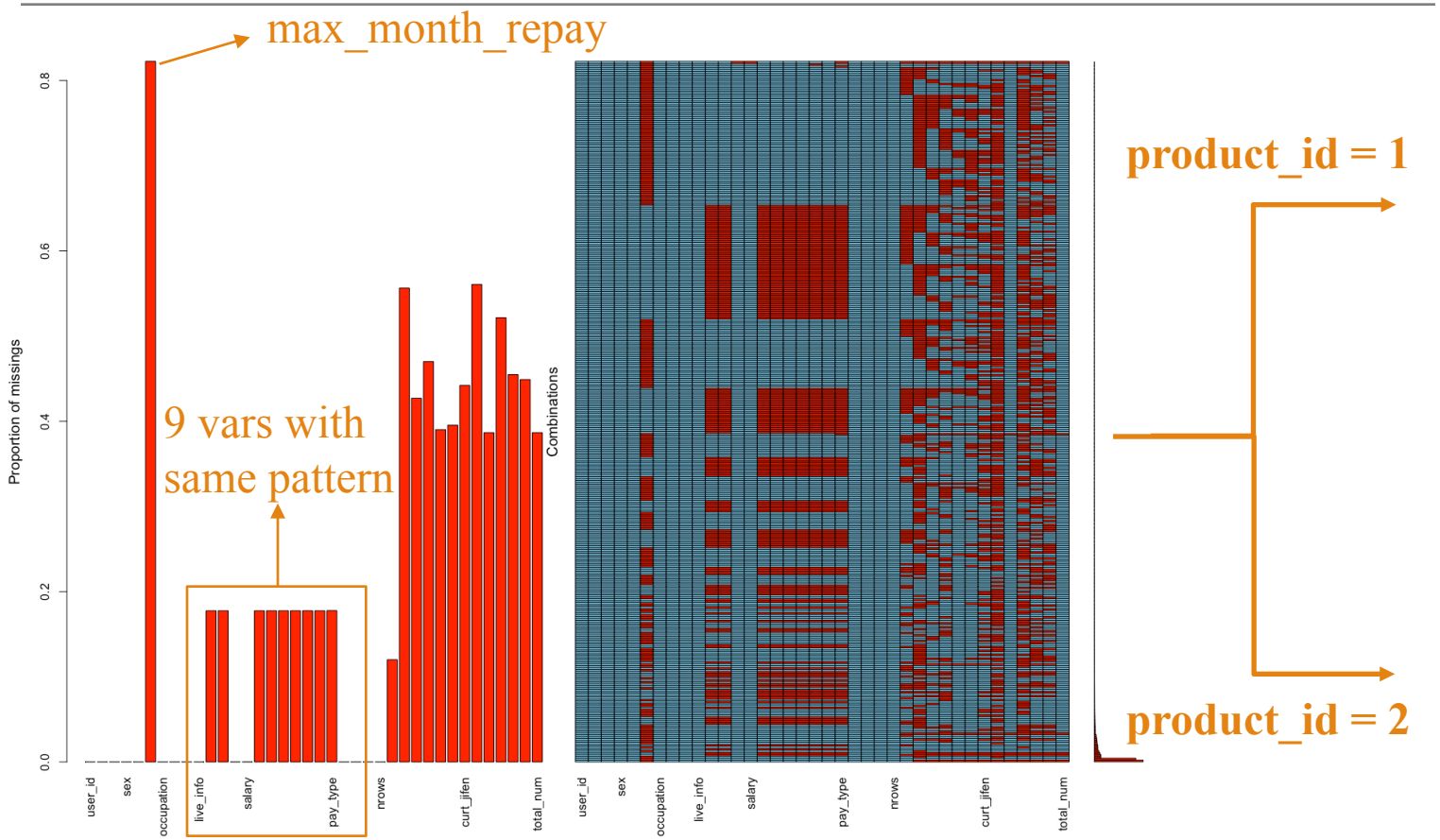
! The type of duplication is related to the value of 'tm_encode' .

Duplication I(Add new information): 'tm_encode' is larger

Duplication II(Revise old information): 'tm_encode' is smaller

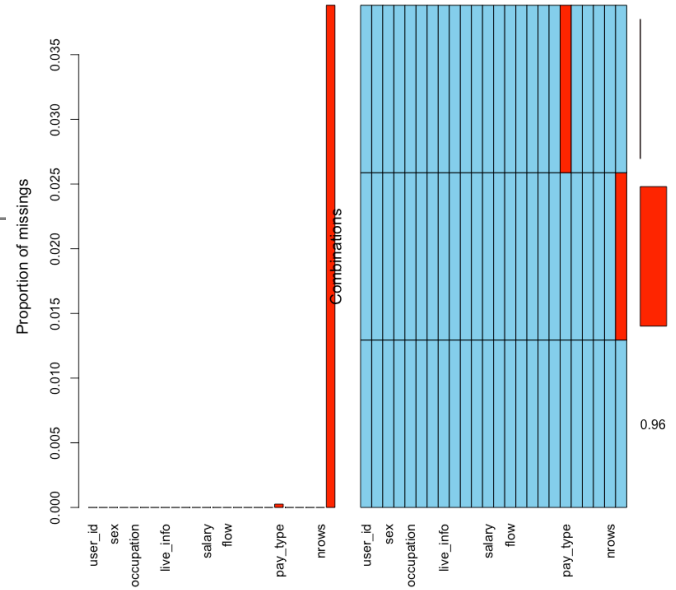
3. EDA

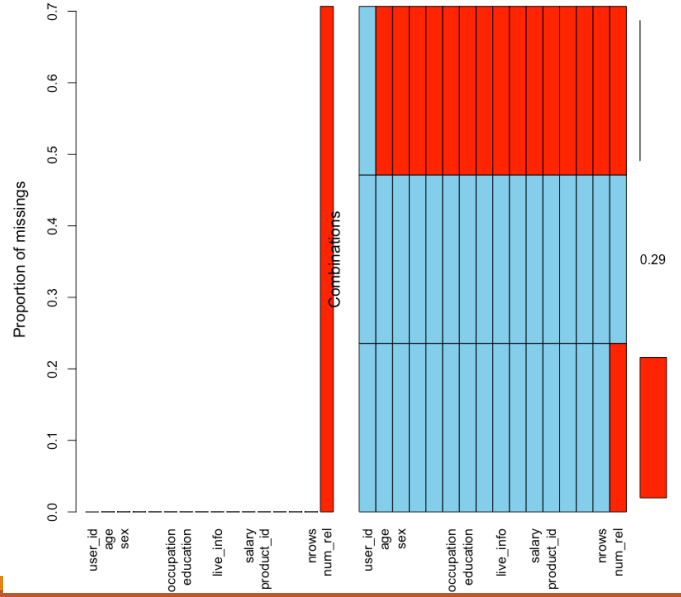
3.2 Missingness: I found that ‘user_info’ has two obvious patterns of missingness for different product_id (1 or 2).



product_id = 1

product_id = 2

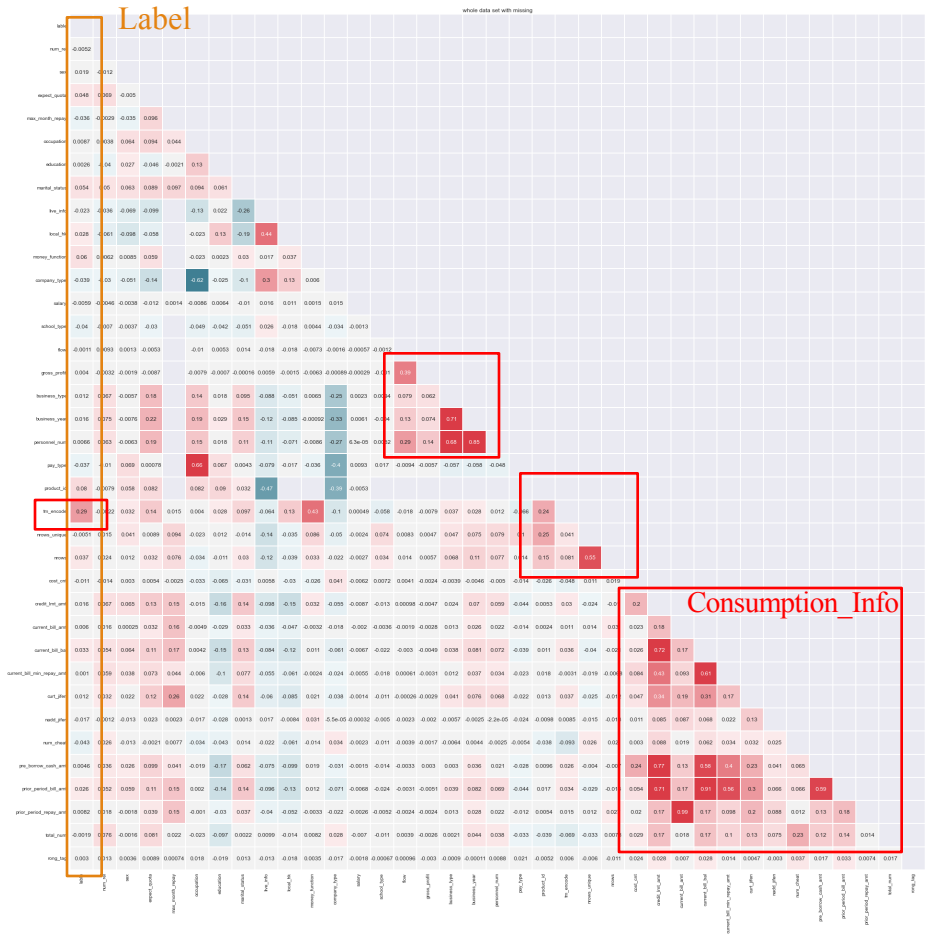




3. EDA

3.3 Correlation of Features :

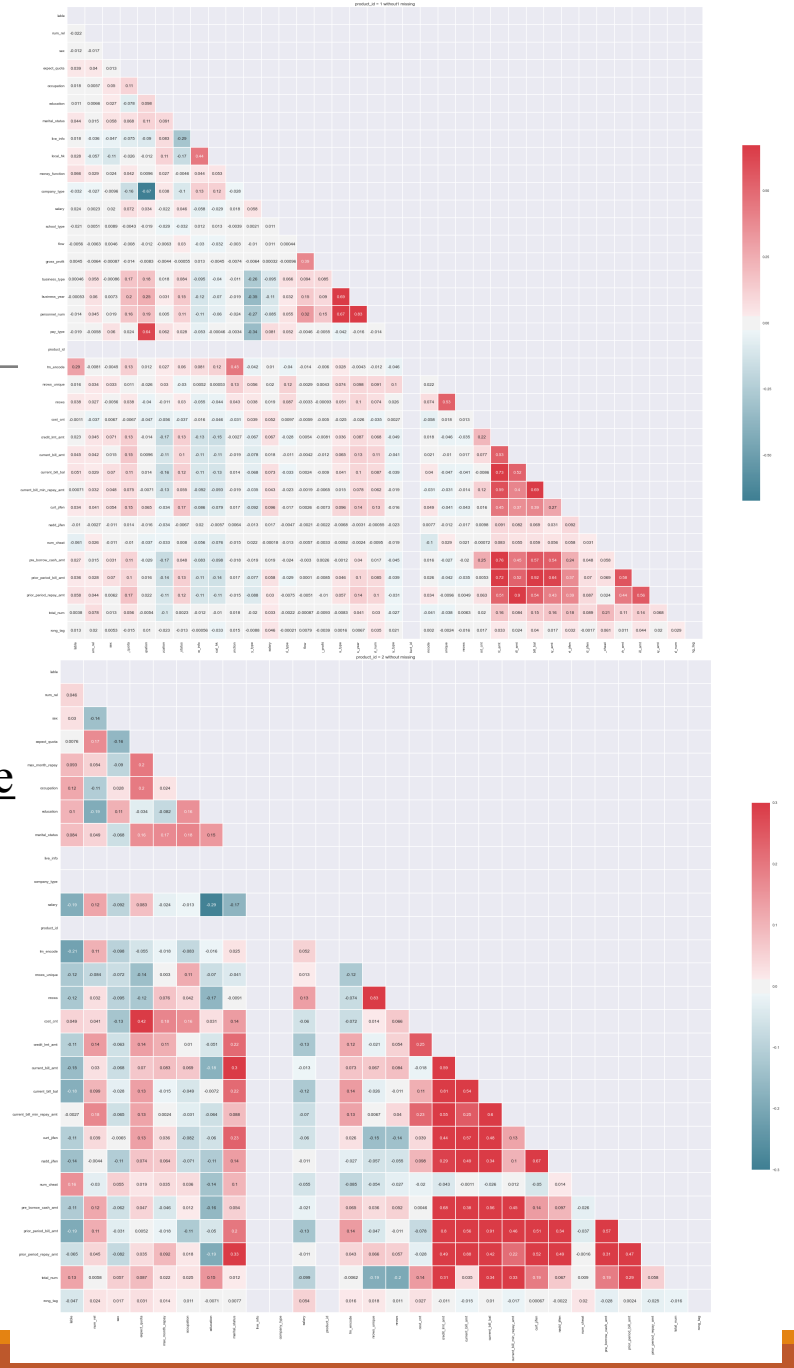
Variables in ‘consumption_Info’ are high correlated with each other.
Importance of variables is different for different ‘product_id’s.



product_id = 1

complete case

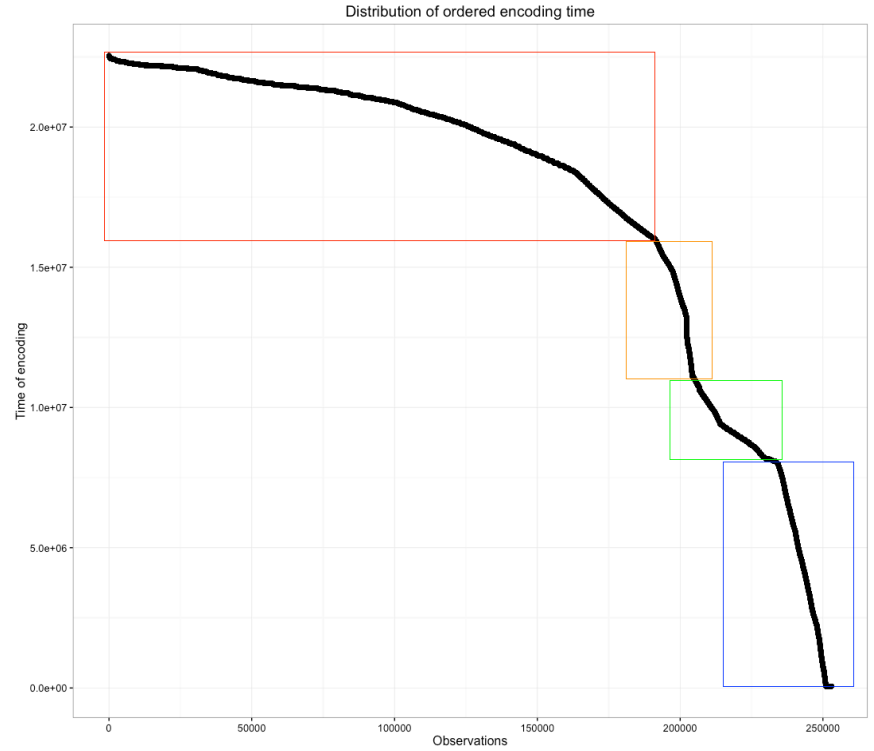
product_id = 2

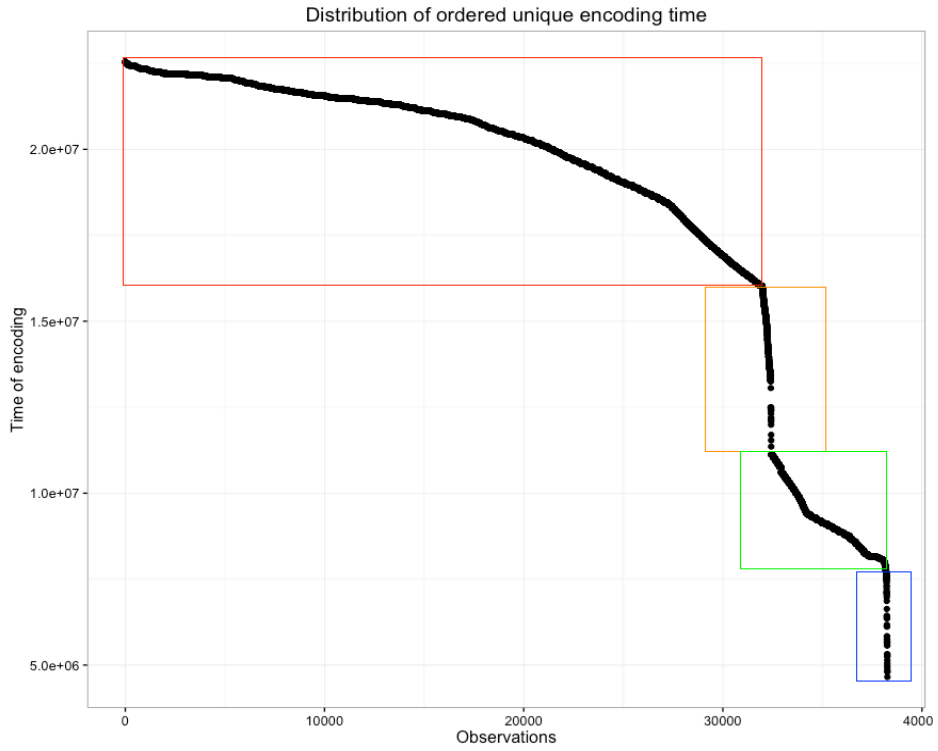


4. Feature Engineering

4.1 User_id Duplication:

Selecting observations with the largest ‘tm_encode’, which means selecting newer and more complete information.



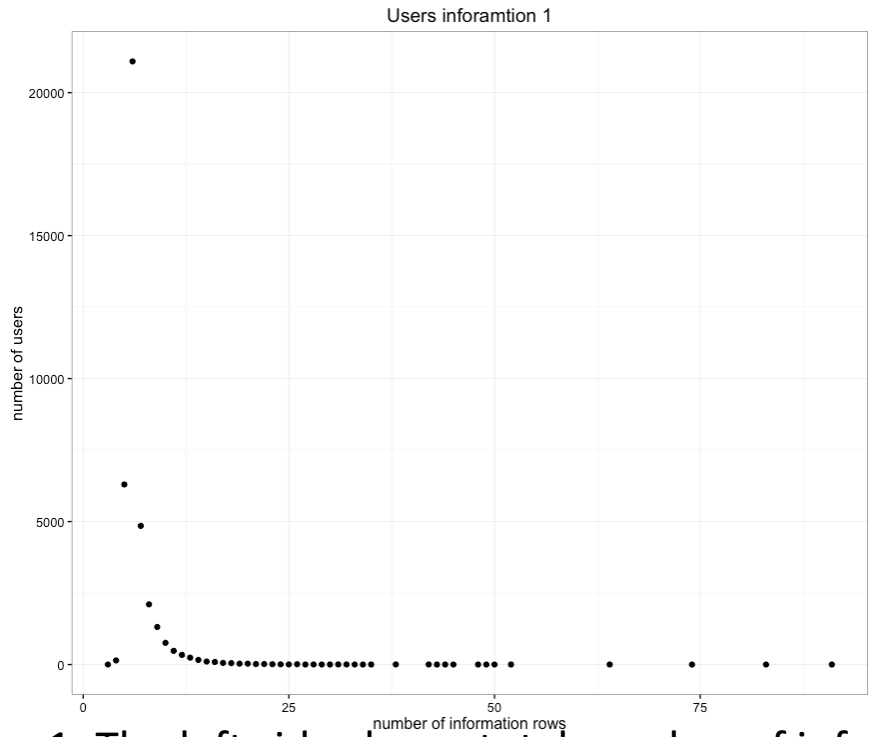


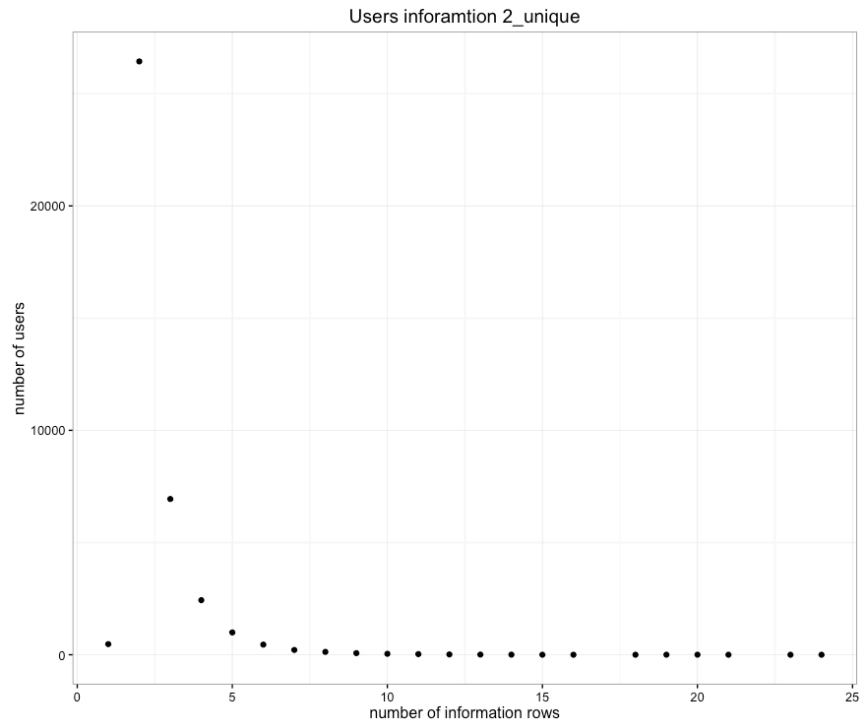
Four patterns in left side and right side are very closed, which means this method retains most information of ’tm_encode’.

4. Feature Engineering

4.1 User_id Duplication:

I keep information about the number of information changes with two dummy variables





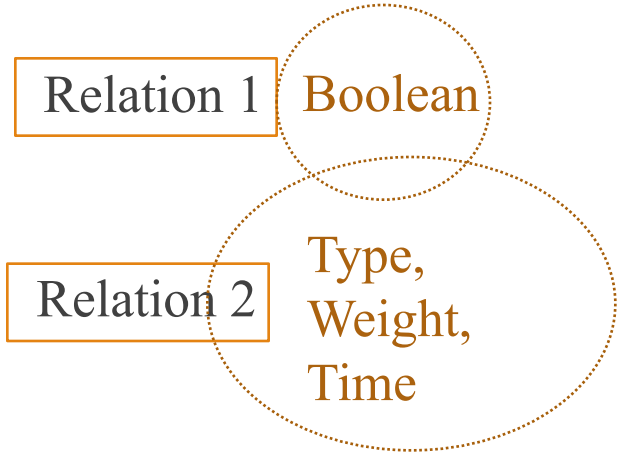
Dummy1: The left side shows total number of information changes for every ‘user_id’, including users with new information (variables besides ‘tm_encode’ changed) and without new information (Only ‘tm_encode’ changed).

Dummy2: The right side shows the number of information changes from users with new information (variables besides ‘tm_encode’ changed)

4. Feature Engineering

4.2 Missingness & Incomplete Information of Users

I applied different methods for different datasets and different models.



Transform Boolean value into numeric value by recording the number of relations for every user

Transform relation value into numeric and categorical value by a weighted analysis. (A piecewise equation)

For logistic regression and Random Forest, I remove this dataset due to the lack of a large percentage of information
For XGboost, I keep this dataset, and apply a ‘average’ method which is similar to ‘K-means to fix the ‘user_id’ duplication problem.

From the results of models and correlation map in EDA, I think this data is not much useful, but I keep it in models and impute missing value with simple methods such as mean() & selecting the most frequent value.

5. Modeling

Model 1: product_id = 1

- (school_type, flow, gross_profit, bussiness_type, bussiness_year, personal_num, pay_type, money_function, local_hk) [Information not existed, ‘NA’]

- (live_info, company_type) [Information missed, all ‘0’]

Model 2: product_id = 2

- max_money_repay [information not existed, ‘NA’]

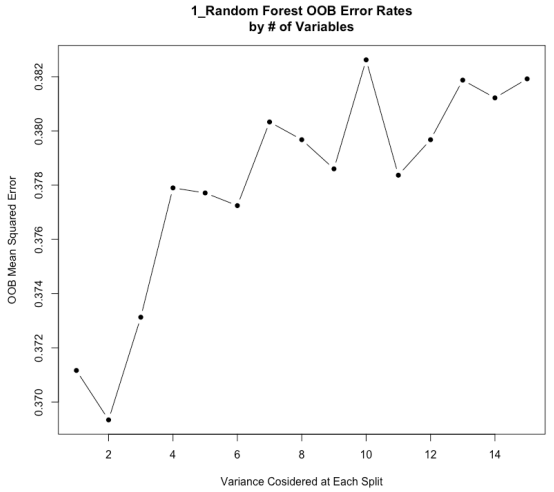
5. Modeling

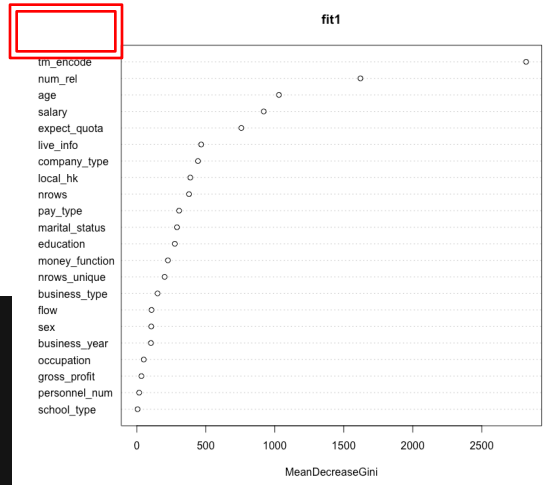
5.1 Random Forest

Model 1:
tree = c(500, 1000, 5000)
mtry = c(1:15)

```
# tree = 500
oob.err1 = numeric(15)
for (mtry in 1:15) {
  fit1 = randomForest(lable ~ .-user_id -product_id, data = train_1, mtry = mtry)
  oob.err1[mtry] = fit1$err.rate[500]
  cat('We are performing iteration', mtry, '\n')
}

plot(1:15, oob.err1, pch = 16, type = 'b',
     xlab = 'Variance Cosidered at Each Split',
     ylab = 'OOB Mean Squared Error',
     main = '1_Random Forest OOB Error Rates\nby # of Variables')
# variane important:
varImpPlot(fit1)
oob.err1[1]
```






```
# mtry = 2 # we believe this model is the most appropriate one
fit0 <- randomForest(lable ~ .-user_id -product_id, data = train_1, mtry = 2, ntree = 5000)
fit0
varImpPlot(fit0)
```

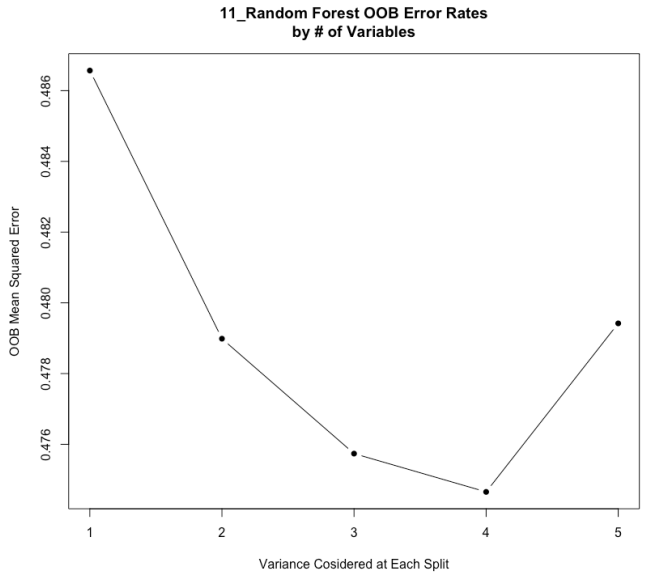
5. Modeling

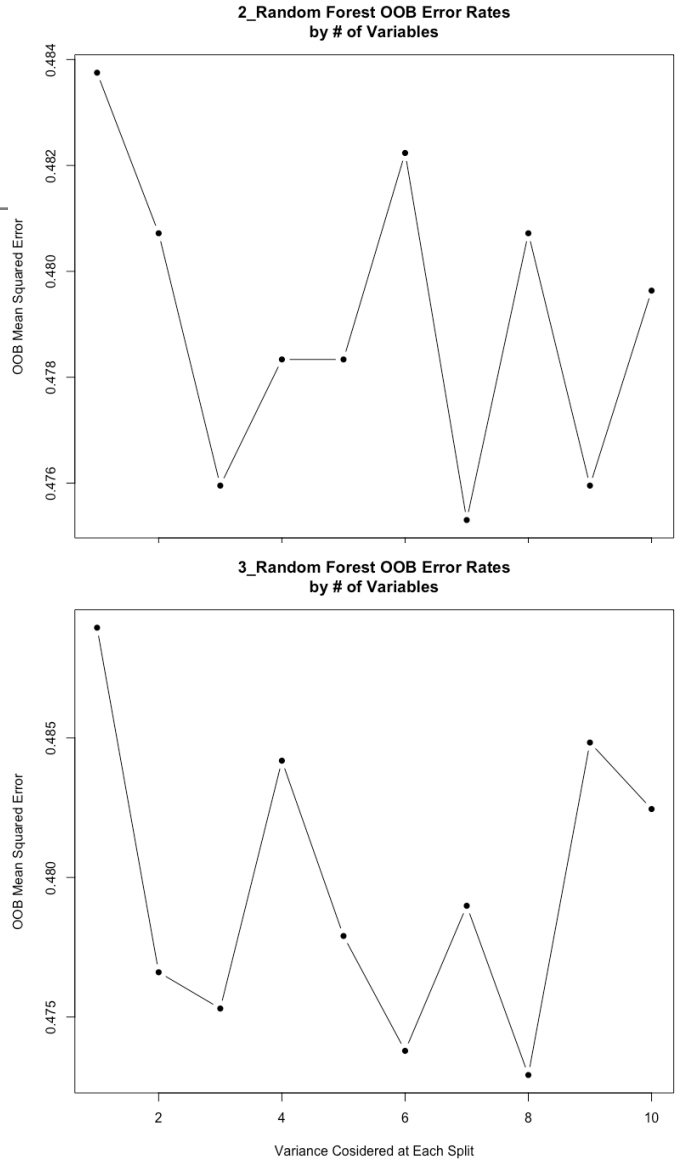
5.1 Random Forest:

Model 2:

tree = c(500, 1000, 5000, 10000, 20000)

mtry = c(1:15)






```
##### best model ######
# mtry = 3
fitt0 <- randomForest(lable ~ .-user_id -product_id, data = train_2, mtry = 3, ntree = 10000)
fitt0
varImpPlot(fitt0)
```

5. Modeling

5.2 XGBoost:

```
############## model 1: product_type = 1 ##############
xgb_data2.0 <- sapply(train_1[, -c(1, 2)], as.numeric)
xgb_train2.0 <- xgb.DMatrix(data = xgb_data2.0, label = train_1[, 2], missing = NA)

xgb_cv_model2.0 <- xgb.cv(data = xgb_train2.0, max_depth = 7, eta = .01, sub_sample = .9,
                          nround = 500, eval_metric = 'auc', objective = 'binary:logistic',
                          early.stop.round = 200, prediction = T, nfold = 5)
```

nrounds, eta, max_depth
early.stop.round is used to avoid overfitting

Model 1

```
# seed = 105, nround = 300, best = 300: [299] train-auc:0.844352+0.007536 test-auc:0.667742+0.001953
# seed = 105, nround = 500, best = 167: [166] train-auc:0.819525+0.004736 test-auc:0.664645+0.009051
# seed = 105, nround = 1000, best = 328: [327]  train-auc:0.841931+0.006249 test-auc:0.666858+0.002319
# seed = 105, nround = 2000, best = 318: [317]  train-auc:0.844811+0.008292 test-auc:0.666135+0.005498
# seed = 105, nround = 3000, best = 298: [297]  train-auc:0.842452+0.009338 test-auc:0.664492+0.002666

### max_depth = 12, early.stop.round = 500
# seed = 105, nround = 3000, best = 352: [351]  train-auc:0.935483+0.008792 test-auc:0.658165+0.004535
### max_depth = 15, early.stop.round = 500
# seed = 105, nround = 3000, best = 426: [425]  train-auc:0.987734+0.002690 test-auc:0.656741+0.007574

### max_depth = 15, early.stop.round = 200
# seed = 105, nround = 3000, best = 483: [482]  train-auc:0.989436+0.003427 test-auc:0.655728+0.006508

### max_depth = 10, early.stop.round = 200
# seed = 105, nround = 3000, best = 181: [180]  train-auc:0.858964+0.010110 test-auc:0.662397+0.003953
# seed = 105, nround = 1000, best = 302: [301]  train-auc:0.873834+0.006803 test-auc:0.662872+0.006273

### max_depth = 8, early.stop.round = 200
# seed = 105, nround = 1000, best = 200: [199]  train-auc:0.795741+0.002447 test-auc:0.666972+0.005977

### max_depth = 7, early.stop.round = 200
# seed = 105, nround = 500, best = 269: [268] train-auc:0.770949+0.004435 test-auc:0.670817+0.009324
# seed = 105, nround = 500, best = 300, sub_sample = .8: [299]  train-auc:0.775846+0.001979 test-auc:0.670077

### max_depth = 6, early.stop.round = 200
# seed = 105, nround = 1000, best = 342: [341]  train-auc:0.749814+0.004327 test-auc:0.669340+0.005324
```

5. Modeling

5.2 XGBoost:

```
############## model 2: product_type = 2 ###############
xgb_data2.1 <- sapply(train_2[, -c(1, 2, 10, 11)], as.numeric)
xgb_train2.1<- xgb.DMatrix(data = xgb_data2.1, label = train_2[, 2], missing = NA)

xgb_cv_model2.1 <- xgb.cv(data = xgb_train2.1, max_depth = 8, eta = .01, sub_sample = .7,
                          nround = 1000, eval_metric = 'auc', objective = 'binary:logistic',
                          early.stop.round = 200, prediction = T, nfold = 5)
```

```
# set.seed = 105
### max_depth = 12, early.stop.round = 200
# nround = 3000, best = 280: [279]  train-auc:0.986048+0.002023 test-auc:0.666603+0.011616
### max_depth = 11, early.stop.round = 200
# nround = 1000, best = 208: [207]  train-auc:0.965220+0.006570 test-auc:0.662185+0.020415
### max_depth = 10, early.stop.round = 200
# nround = 1000, best = 367: [366]  train-auc:0.965944+0.005171 test-auc:0.665819+0.003264
### max_depth = 9, early.stop.round = 200
# nround = 1000, best = 237: [236]  train-auc:0.924071+0.003180 test-auc:0.672658+0.015114
# nround = 1000, sub_sample = .8, best = 205: [204] train-auc:0.921374+0.005797 test-auc:0.669313+0.013268

### max_depth = 8, early.stop.round = 200
# nround = 1000, best = 121: [122]  train-auc:0.871667+0.007527 test-auc:0.673707+0.019273
# nround = 1000, sub_sample = .8, best = 246: [245] train-auc:0.896504+0.005707 test-auc:0.674522+0.012812
# nround = 1000, sub_sample = .7, best = 126: [125] train-auc:0.869131+0.010227 test-auc:0.677087+0.006542 ***
# nround = 1000, sub_sample = .6, best = 246: [245] train-auc:0.896398+0.007088 test-auc:0.676580+0.017664

### max_depth = 7, early.stop.round = 200
# nround = 1000, best = 186: [185]  train-auc:0.855022+0.001725 test-auc:0.680919+0.018301
```

Model 2

Note:
I also applied Logistic Regression to this problem, but it's much worse than the combinations of these two models.

6. Ranking

比赛实时排行榜

初赛结束时间：

2016.10.8 24:00

您最近一次预测得分为0.5798（2016-09-14），历史最高预测得分为0.5798，目前排名第78位

排名	预测得分	参赛队	赛区	所在学校
1	0.6184	青岛黄渤	上海赛区	青岛大学
2	0.6180	海淀吴彦祖	北京赛区	北京大学
3	0.6160	微博搞笑排行榜	广州赛区	中山大学
4	0.6148	大连宁泽涛	广州赛区	大连交通大学
5	0.6143	alone	广州赛区	云南大学

比赛实时排行榜

初赛结束时间：

2016.10.8 24:00

您最近一次预测得分为0.5667（2016-09-18），历史最高预测得分为0.5798，目前排名第86位

排名	预测得分	参赛队	赛区	所在学校
1	0.6199	青岛黄渤	上海赛区	青岛大学
2	0.6181	海淀吴彦祖	北京赛区	北京大学
3	0.6160	微博搞笑排行榜	广州赛区	中山大学
4	0.6148	大连宁泽涛	广州赛区	大连交通大学
5	0.6143	alone	广州赛区	云南大学

比赛实时排行榜

初赛结束时间：

2016.10.8 24:00

您最近一次预测得分为0.5882（2016-09-19），历史最高预测得分为0.5882，目前排名第83位

排名	预测得分	参赛队	赛区	所在学校
1	0.6199	青岛黄渤	上海赛区	青岛大学
2	0.6181	海淀吴彦祖	北京赛区	北京大学
3	0.6160	微博搞笑排行榜	广州赛区	中山大学
4	0.6148	大连宁泽涛	广州赛区	大连交通大学
5	0.6145	暨南苏炳添	广州赛区	暨南大学

Ranking:
83/600+

Best Ranking:
78/600+

Score: 0.5798 → 0.5667 → 0.5882 →

Model: Two models | One models | Combinations of two models | Ensemble Modeling

Thank you!

You are very welcomed to discuss any questions
or kindly offer suggestions at:

Address: Ellendale, Los Angeles, CA
Call: +1-213-245-3029
Email: shuliu@usc.edu
Linkedin: https://www.linkedin.com/in/shu-liu
wechat: 2235127014

Master of Financial Engineering
@ University of Southern California