Lu Guannan
20454477

2017/12/1

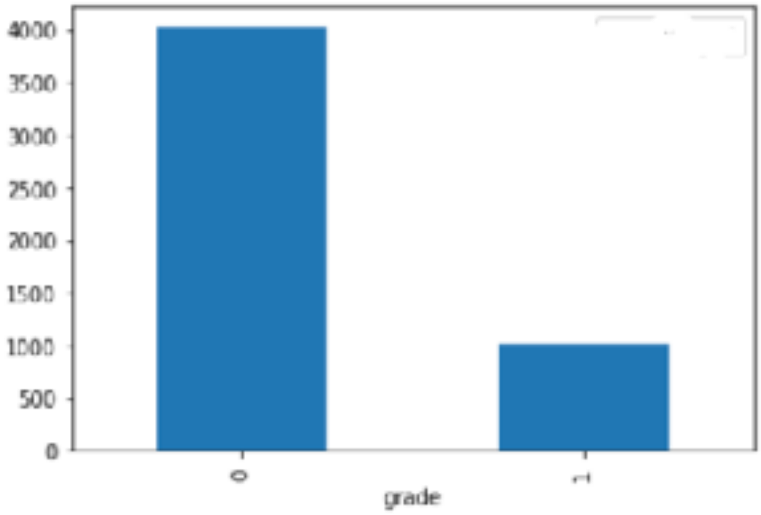
5002 Ass3 Report

1. **Classification Task (60 marks)**
    1. Training Enviroment

       Python 2.7, sklearn, numpy, pandas, collections, imblearn

    2. Features engineer

- Unbalance Data



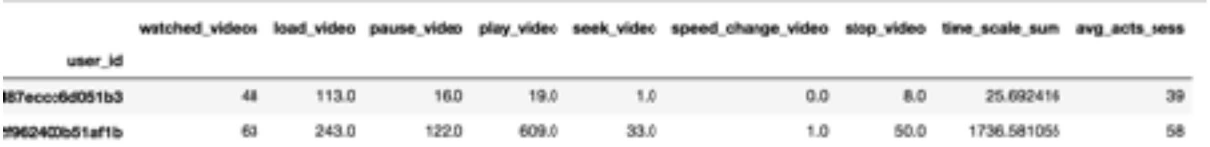
1. Fulling NaN

Column	NaN Percentage
new_speed	0.993472
old_speed	0.993472
new_time	0.837754
old_time	0.837754

Because these features have value only when certain actions occur, I fill 0 to this feature.

2.Generate features :

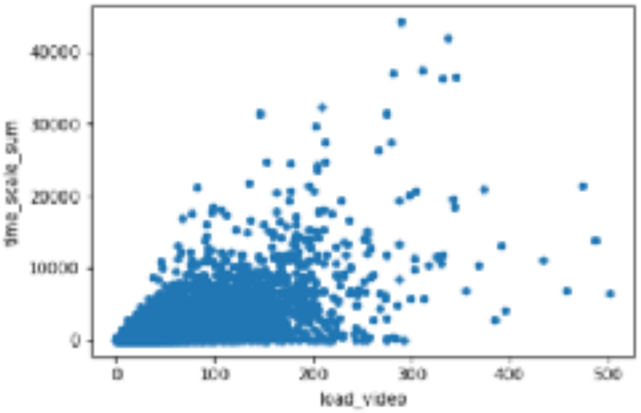
Feature	Meaning
watched_videos	Number of watched videos per person
session	Number of used sessions per person
load_video	Number of actions for every person
pause_video	Number of actions for every person
play_video	Number of actions for every person
seek_video	Number of actions for every person
speed_change_video	Number of actions for every person
stop_video	Number of actions for every person
time_scale_sum	Sum of skipped time during watched video for every user
avg_acts_sess	Average number of actions per person take by using one session

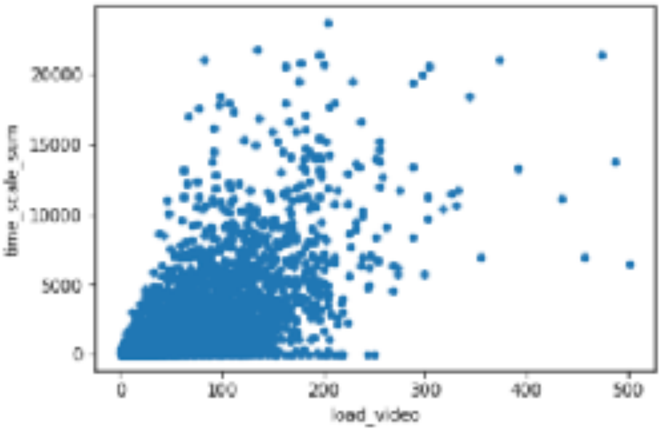


	watched_videos	load_video	pause_video	play_video	seek_video	speed_change_video	stop_video	time_scale_sum	avg_acts_sess
user_id									
87ecc06d051b3	48	113.0	16.0	19.0	1.0	0.0	8.0	25.692416	39
f962400b51af1b	63	243.0	122.0	609.0	33.0	1.0	50.0	1736.581055	58

3. Outliers detection

I use LocalOutlierFactor packet to calculate the LOF , and then cut 0.08 outliers of all samples. Time_scale_sum max value is reduced by LOF.





4. Resample

   Class to perform over-sampling using SMOTE and cleaning using ENN.

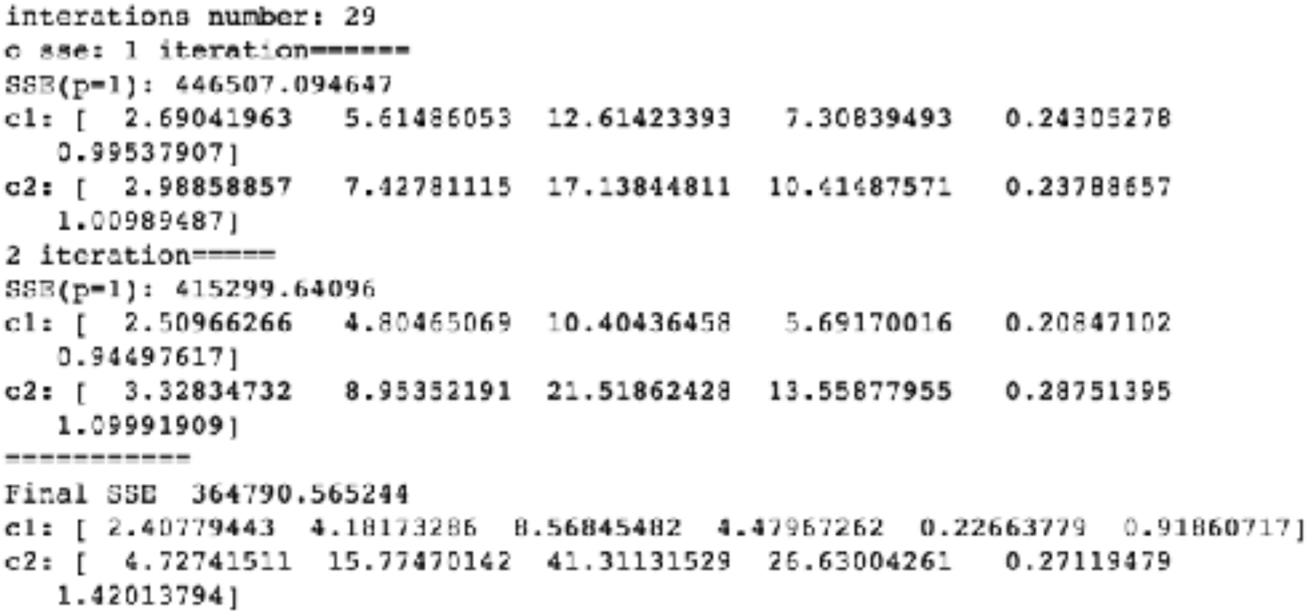
   Combine over- and under-sampling using SMOTE and Edited Nearest Neighbors.

5. Model

   Using **RandomForest** to classify the unbalanced data.

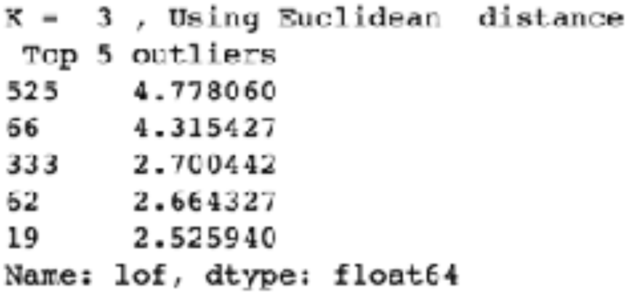
6. Reference

2. Fuzzy clustering with EM algorithm

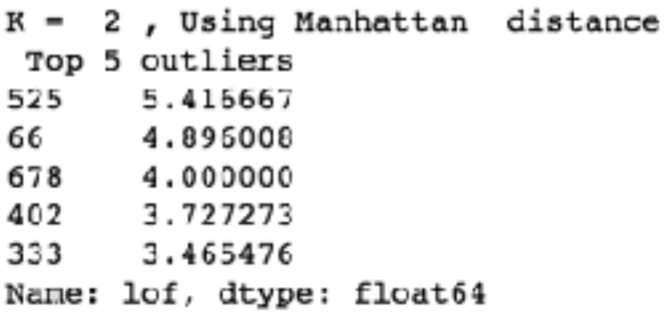


```
interations number: 29
c sse: 1 iteration======
SSE(p=1): 446507.094647
c1: [  2.69041963   5.61486053  12.61423393   7.30839493   0.24305278
   0.99537907]
c2: [  2.98858857   7.42781115  17.13844811  10.41487571   0.23788657
   1.00989487]
2 iteration=====
SSE(p=1): 415299.64096
c1: [  2.50966266   4.80465069  10.40436458   5.69170016   0.20847102
   0.94497617]
c2: [  3.32834732   8.95352191  21.51862428  13.55877955   0.28751395
   1.09991909]
===========
Final SSE  364790.565244
c1: [ 2.40779443  4.18173286  8.56845482  4.47967262  0.22663779  0.91860717]
c2: [  4.72741511  15.77470142  41.31131529  26.63004261   0.27119479
   1.42013794]
```

3. Outlier detection with LOF



```
K =   3 , Using Euclidean  distance
 Top 5 outliers
525     4.778060
66      4.315427
333     2.700442
62      2.664327
19      2.525940
Name: lof, dtype: float64
```



```
K =  2 , Using Manhattan  distance
 Top 5 outliers
525    5.416667
66     4.896008
678    4.000000
402    3.727273
333    3.465476
Name: lof, dtype: float64
```