作法

1. 從training data中把「同樣USER-ID」評書的紀錄收集起來
    - $U_i = \{r_1, r_2, \ldots, r_j\}$
2. 計算各種統計量（平均、最大最小、偏態、峰態、10分位數）
3. 把$U_i$序列壓扁當成該User的特徵
4. 沒有資料的User特徵用中位數填入
    - 可能看過零本書、都是implict rating、users.csv根本沒描述、...等
5. （忽略ISBN）
6. train lightGBM
    - 參數全部用預設
    - 微軟出的gradient boost decsision tree套件，anaconda沒有要自己裝
7. 輸出track2預測、四捨五入到整數位當成track1預測

結論

1. 大部分使用者只評1本書（之後另寫一篇）
    - 沒算變異數是因為scipy呼叫了長度1的序列會掛掉XD
2. lightGBM train得比sklearn.ensemble.GradientBoostingClassifier快得多
    - 有很多錯誤函數可以選，之後再研究
3. lb score看來使用者評分的"scale"是有點重要（專門評低分的人vs都給8,9,10分的人）

可能的改進

1. 調lightGBM的參數
2. 對ISBN做簡單分析
3. 對user資料做一些視覺化（i.e. $U_i$）

---

# DataLoader / Imports

In [1]:

```
import pandas as pd
import numpy as np
import scipy
from learning2read.utils import DataLoader
def Data(key,**kwargs):
    return DataLoader(r"/Users/qtwu/Downloads/data").load(key,**kwargs)
```

In [2]:

```
raw_user=Data("user")
raw_train=Data("btrain")
raw_test=Data("btest")
raw_submit=Data("submit",index_col=None,header=None)
raw_user.shape, raw_train.shape, raw_test.shape, raw_submit.shape
```

```
/Users/qtwu/Downloads/data/users.csv
/Users/qtwu/Downloads/data/book_ratings_train.csv
/Users/qtwu/Downloads/data/book_ratings_test.csv
/Users/qtwu/Downloads/data/submission.csv
```

Out[2]:

```
((278858, 3), (260202, 3), (173469, 2), (173469, 1))
```

# Build User Features By "ratings"

In [3]:

```
user_dict={}
def new_user(uid):
    global user_dict
    user_dict[uid]={'nbook':0,'ratings':[]}
for r in raw_train.to_dict('record'):
    uid=r['User-ID']
    if not user_dict.get(uid):
        new_user(uid)
    user_dict[uid]['nbook']+=1
    user_dict[uid]['ratings'].append(r['Book-Rating'])
```

**Slow** caculating statistics (moment/quantile)

In [4]:

```python
import scipy.stats
stat_name=['mode','tmean','tmin','tmax','skew','kurtosis']
def gen(uid,r):
    result={
        'User-ID':uid,
        'nbook':r['nbook']
    }
    for name in stat_name:
        result["rating_"+name]=eval("scipy.stats.%s(r['ratings'])"%name)
    for v in range(9):
        q=(v+1)*10
        result['rating_q%d'%(v+1)]=np.percentile(r['ratings'],q)
    return result
df_user_rate=pd.DataFrame([gen(uid,r) for uid,r in user_dict.items()])
df_user_rate['rating_mode']=df_user_rate['rating_mode'].apply(lambda r:r[0][
df_user_rate.sample(3)
```

Out[4]:

| | User-ID | nbook | rating_kurtosis | rating_mode | rating_q1 | rating_q2 | rating_q3 | rating_q4 | rating_ |
|---|---|---|---|---|---|---|---|---|---|
| **12437** | 5b4f561f42 | 15 | -0.343264 | 7 | 6.0 | 6.8 | 7.0 | 7.0 | 7 |
| **17102** | 7fcfcfb714 | 17 | -1.603337 | 5 | 4.6 | 5.0 | 5.0 | 5.8 | 7 |
| **21193** | d4f08d3783 | 4 | -1.000000 | 5 | 5.9 | 6.8 | 7.7 | 8.2 | 8 |

# Prepare Training Data

In [5]:

```python
def rating_merge(rating,user): # only users.csv, books disposed
    df=rating
    df=df.merge(user,on='User-ID',how='left')
    df=df.drop(['User-ID','ISBN'],1)
    df=df.fillna(df.median()) # fill with median
    return df
df_train=rating_merge(raw_train,df_user_rate)
df_train.sample(3)
```

Out[5]:

| | Book-Rating | nbook | rating_kurtosis | rating_mode | rating_q1 | rating_q2 | rating_q3 | rating_q4 | rating_q5 |
|---|---|---|---|---|---|---|---|---|---|
| **243737** | 7 | 8 | 0.721253 | 9 | 6.1 | 7.0 | 7.1 | 7.8 | 8.0 |
| **186187** | 8 | 28 | -0.668800 | 5 | 4.0 | 5.0 | 5.0 | 5.0 | 7.0 |
| **178929** | 9 | 175 | -0.432006 | 8 | 7.0 | 7.0 | 7.0 | 8.0 | 8.0 |

# Train LightGBM

(may be slow)

In [6]:

```
# from sklearn.ensemble import GradientBoostingClassifier,RandomForestRegres
import lightgbm as lgb
import datetime

x_train=df_train.iloc[:,1:]
y_train=np.ravel(df_train.iloc[:,:1])

# model=RandomForestRegressor(
#     500,
#     max_features='sqrt',
#     verbose=1,
#     n_jobs=-1,
# )

# model=GradientBoostingClassifier(
#     verbose=1,
#     criterion='mae',
#     n_estimators=10,
# )

model=lgb.LGBMRegressor(objective='regression')

st=datetime.datetime.now()
model.fit(x_train, y_train)
print(datetime.datetime.now()-st)
```

```
0:00:04.380443
```

# Prepare Testing Data (for Submission)

In [7]:

```
x_test=rating_merge(raw_test,df_user_rate)
```

In [8]:

```
def output_test(est_name="empty"):
    y_test_predict=model.predict(x_test)
    y_test_predict=pd.DataFrame(y_test_predict)
    y_test_predict.describe()

    df_output=raw_submit.iloc[:,:]
    df_output.iloc[:,0]=y_test_predict

    df_output2=df_output.iloc[:,:]
#     df_output2=df_output2.transform(lambda x: (x/10)**1.5*10 )
    df_output2=df_output2.round(1)
    df_output2.to_csv("track2_%s.csv"%est_name,header=None,index=None)
    print(df_output.describe())
    df_output=df_output.round()
    df_output=df_output.astype('int32')
    df_output.to_csv("track1_%s.csv"%est_name,header=None,index=None)
    print(df_output.describe())
# output_test("gbm_default_1.5down")

output_test("gbm_default_param")

# RESULT #
"""
track1:
b04303128   2018-05-30 09:45:23
Banana
BananaBanana    1.273607
"""
pass
```

```
                   0
count  173469.000000
mean        7.629817
std         1.071980
min         0.994106
25%         7.140688
50%         7.687896
75%         8.250696
max         9.999731
                   0
count  173469.000000
mean        7.683505
std         1.126445
min         1.000000
25%         7.000000
50%         8.000000
75%         8.000000
max        10.000000
```

In [10]:

```
# :p
import homework
from homework import *
reload(homework)
pass
```