01-movie-rating-data-exploration

2017년 4월 13일

0.1 별점 데이터 탐색

이것으로 추천이 되겠다 안되겠다는 사전에 판단을 하기는 어렵다… 그러나 추천 시스템을 만들어 보고 그 결과가 잘 안나오는것 같다라고 판단할때 원인을 추론하기 용이하기 때문에 이 정도는 알고 가야 서비스 분석에도 도움이 된다.

유저당 별점 매긴 수

- mean, median, skew, 분포

영화별 별점 받은 수

- mean, median, skew, 분포
- 영화 개봉기간에 따른 별점수 분포

별점 예측 결과 분석

- 별점 매긴 수별 정확도
-

0.2 별점을 많이 맞은 수별 정확도

- seaborn package install
- http://stanford.edu/~mwaskom/software/seaborn/
- http://stanford.edu/~mwaskom/software/seaborn/installing.html#installing
- 설치가 안되어 있다면 그래프 부분만 제외하고 실행가능

In [1]:
```
# coding: utf-8

import pandas as pd
import numpy as np
from matplotlib import rcParams
import seaborn as sns
import matplotlib.pyplot as plt
from collections import defaultdict
from datetime import datetime
import matplotlib.patches as mpatches
import matplotlib
from __future__ import print_function
%matplotlib inline

rcParams['font.family'] = 'NanumGothic'
rcParams.update({'font.size': 12})
matplotlib.style.use('ggplot')
sns.set_style("whitegrid")
```

Movie Lens 데이터 로드 http://grouplens.org/datasets/movielens/

In [2]:
```
def movieLensDataLoad(type):
    ## user 영화 별점 data
    ratings = pd.read_csv("/Users/youngseoklee/Dropbox/fc-recsys-school-master/ch3/reso

    ## movie meta(타이틀,장르) data
    movies = pd.read_csv("/Users/youngseoklee/Dropbox/fc-recsys-school-master/ch3/resou

    ## user가 영화에 tag를 기입한 data
    tags = pd.read_csv("/Users/youngseoklee/Dropbox/fc-recsys-school-master/ch3/resour
    # tags = pd.read_csv("/Users/goodvc/Documents/data-analytics/movie-recommendation/
    return ( ratings, movies, tags )

#ratings, movies, tags = movieLensDataLoad('ml-20m')
ratings, movies, tags = movieLensDataLoad('ml-latest-small')
```

In [3]:
```
ratings.head()
```

```
Out[3]:    userId  movieId  rating   timestamp
        0       1       31     2.5  1260759144
        1       1     1029     3.0  1260759179
        2       1     1061     3.0  1260759182
        3       1     1129     2.0  1260759185
        4       1     1172     4.0  1260759205
```

```
In [4]: movies.head()
```

```
Out[4]:    movieId                               title  \
        0        1                    Toy Story (1995)
        1        2                      Jumanji (1995)
        2        3             Grumpier Old Men (1995)
        3        4            Waiting to Exhale (1995)
        4        5  Father of the Bride Part II (1995)

                                          genres
        0  Adventure|Animation|Children|Comedy|Fantasy
        1                   Adventure|Children|Fantasy
        2                              Comedy|Romance
        3                        Comedy|Drama|Romance
        4                                      Comedy
```

```
In [5]: tags.head()
```

```
Out[5]:    userId  movieId                     tag   timestamp
        0      15      339  sandra 'boring' bullock  1138537770
        1      15     1955                  dentist  1193435061
        2      15     7478                 Cambodia  1170560997
        3      15    32892                  Russian  1170626366
        4      15    34162              forgettable  1141391765
```

- 별점 데이터 요약통계

```
In [6]: ratings.describe()
```

```
Out[6]:                userId        movieId         rating     timestamp
        count  100004.000000  100004.000000  100004.000000  1.000040e+05
        mean      347.011310   12548.664363       3.543608  1.129639e+09
```

```
std       195.163838   26369.198969       1.058064  1.916858e+08
min         1.000000       1.000000       0.500000  7.896520e+08
25%       182.000000    1028.000000       3.000000  9.658478e+08
50%       367.000000    2406.500000       4.000000  1.110422e+09
75%       520.000000    5418.000000       4.000000  1.296192e+09
max       671.000000  163949.000000       5.000000  1.476641e+09
```

유저당 별점 매긴 수

- mean, median, skew, 분포

```
In [7]: user_rating_count = ratings.groupby(['userId'])['rating'].count()

In [8]: user_rating_count

Out[8]: userId
        1         20
        2         76
        3         51
        4        204
        5        100
        6         44
        7         88
        8        116
        9         45
        10        46
        11        38
        12        61
        13        53
        14        20
        15      1700
        16        29
        17       363
        18        51
        19       423
        20        98
        21       162
        22       220
```

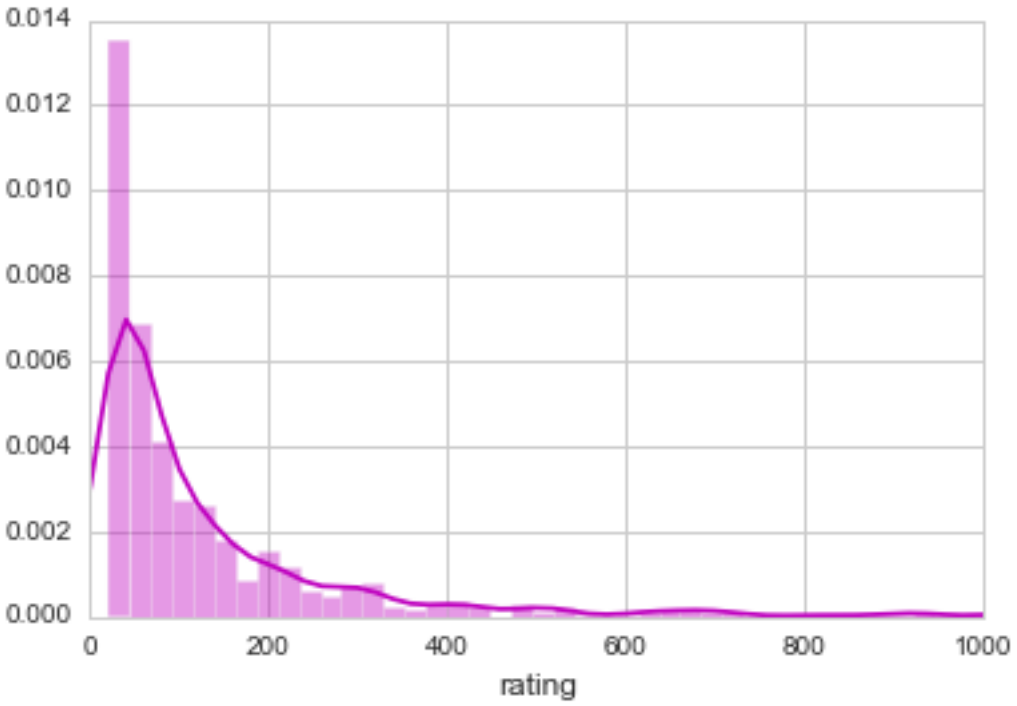
23	726
24	21
25	26
26	172
27	23
28	50
29	22
30	1011
	…
642	36
643	24
644	39
645	30
646	169
647	150
648	256
649	90
650	29
651	20
652	267
653	51
654	626
655	105
656	128
657	20
658	60
659	142
660	92
661	33
662	58
663	26
664	519
665	434
666	40
667	68
668	20

```
669      37
670      31
671     115
Name: rating, dtype: int64
```

```
In [9]: print("유저당 별점 매긴수(mean) %d" % user_rating_count.mean())
        print("유저당 별점 매긴수(median) %d" % user_rating_count.median())
        print("유저당 별점 매긴수(skew) %.3f" % user_rating_count.skew())

        ax = sns.distplot(user_rating_count, color="m",bins=100)
        plt.xlim(0,1000); plt.show();
```

```
유저당 별점 매긴수(mean) 149
유저당 별점 매긴수(median) 71
유저당 별점 매긴수(skew) 4.555
```




```
In [11]: user_rating_count.sort_values()
         user_rating_count.head(100)
```

Out[11]: userId

1	20
498	20
448	20
445	20
444	20
438	20
35	20
399	20
540	20
76	20
337	20
484	20
583	20
319	20
604	20
310	20
325	20
289	20
668	20
657	20
651	20
209	20
296	20
221	20
638	20
485	20
14	20
249	20
477	21
469	21
	..
504	24
180	24
172	24
437	24

```
167     24
643     24
495     25
331     25
269     25
44      25
100     25
556     25
375     25
114     25
377     25
446     25
392     25
116     25
637     25
538     25
170     26
129     26
415     26
663     26
154     26
539     26
246     26
227     26
25      26
181     27
Name: rating, dtype: int64
```

영화별 별점 받은 수

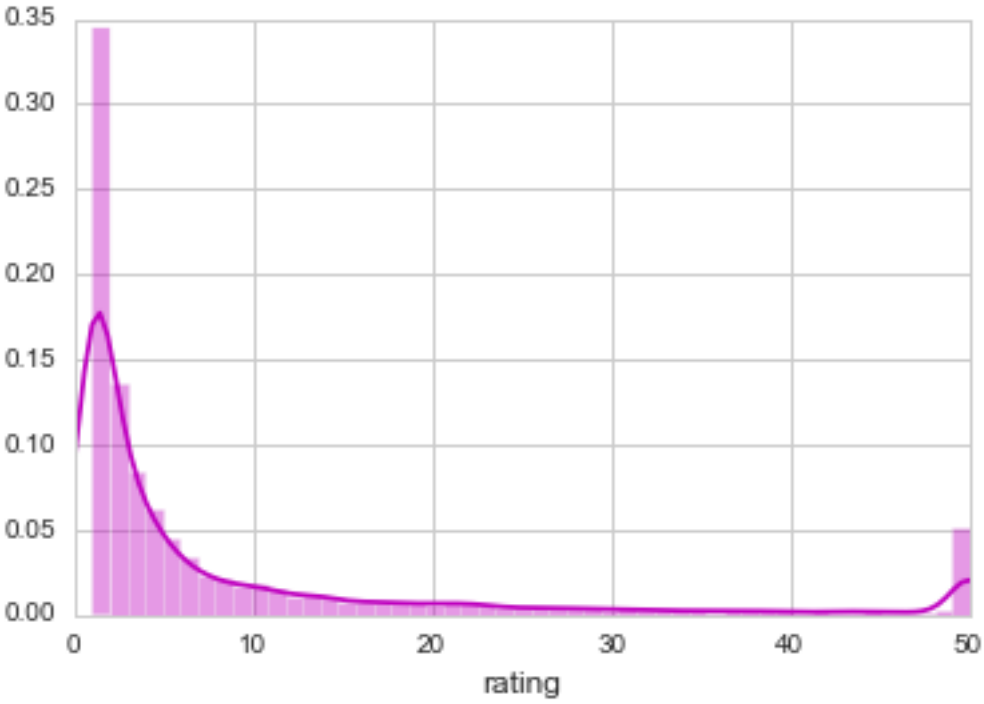
- mean, median, skew, 분포
- 영화 개봉기간에 따른 별점수 분포

```
In [12]: movie_rating_count = ratings.groupby(['movieId'])['rating'].count()
         print("영화당 별점 받은수(mean) %.3f" % movie_rating_count.mean())
         print("영화당 별점 받은수(median) %.3f" % movie_rating_count.median())
```

```
print("영화당 별점 받은수(skew) %.3f" % movie_rating_count.skew())

movie_rating_count[movie_rating_count>50]=50
ax = sns.distplot(movie_rating_count, color="m",bins=50)
plt.xlim(0,50); plt.show()
```

영화당 별점 받은수(mean) 11.031
영화당 별점 받은수(median) 3.000
영화당 별점 받은수(skew) 5.299

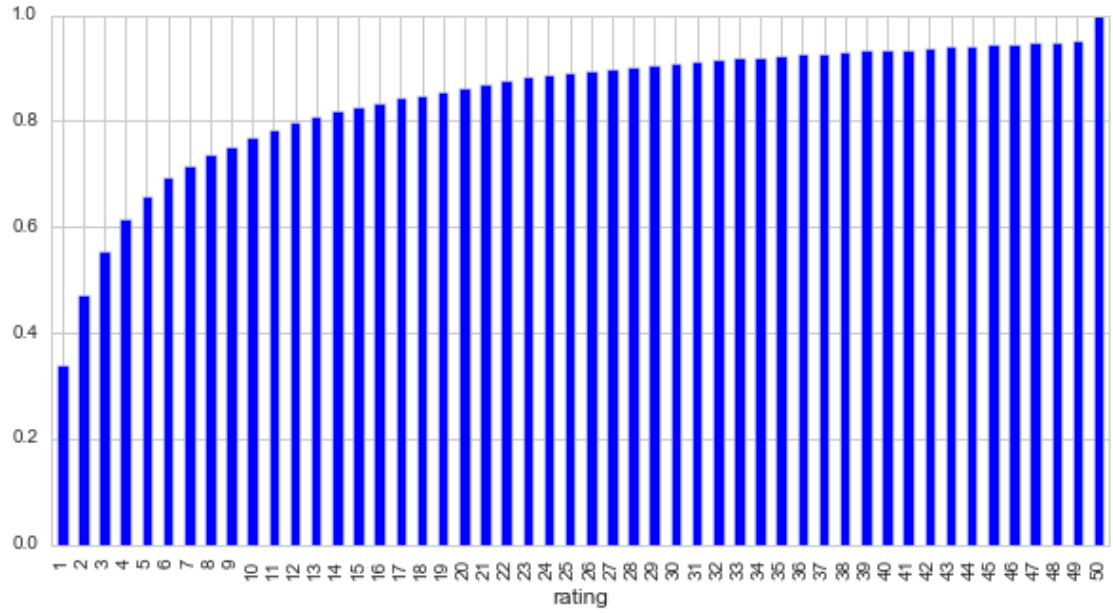



```
In [37]: movie_rating_count.sort_values()
         print("별점 평가수가 5개 미만인 영화 : {0} / {1} = {2}".format(
                 movie_rating_count[movie_rating_count<5].count(),
                 movie_rating_count.count(),
                 float(movie_rating_count[movie_rating_count<5].count())/float(movie_rating_cou
                 )
             )
         ## 별점 평가시 영화의 별점 평가수별 Error율을 측정해 볼 필요가 있음
```

별점 평가수가 5개 미만인 영화 : 5570 / 9066 = 0.614383410545

- 누적합계를 보자

```
In [15]: total = movie_rating_count.count()
         grouped = movie_rating_count.groupby(movie_rating_count).count()
         cumsum = (grouped).cumsum()
         (cumsum/total).plot(kind='bar',figsize=(10,5))
         plt.show()
```




```
In [16]: (grouped*grouped.index).cumsum()
```

```
Out[16]: rating
         1       3063
         2       5467
         3       7732
         4       9932
         5      11917
         6      13747
         7      15189
         8      16685
```

9	18089
10	19709
11	21172
12	22384
13	23840
14	25212
15	26307
16	27395
17	28585
18	29665
19	30900
20	32020
21	33322
22	34752
23	35994
24	36882
25	37732
26	38902
27	39820
28	40744
29	41759
30	42569
31	43561
32	44329
33	45088
34	45768
35	46678
36	47290
37	48215
38	48975
39	49677
40	50477
41	50805
42	51729
43	52460
44	53208

```
45    54063
46    54707
47    55271
48    56039
49    56921
50    79571
Name: rating, dtype: int64
```