BeerRecommendationSystem_x1

September 22, 2015

```
In [19]: import pandas as pd
         import numpy as np
         import pylab as pl
         import os

In [20]: os.chdir(r'E:\Andrey\Stanford\PythonClass')
         df=pd.read_csv("beer_reviews.csv")

In [21]: beer_1,beer_2="Dale's Pale Ale","Fat Tire Amber Ale"

In [22]: beer_1_reviewers=df[df.beer_name==beer_1].review_profilename.unique()
         beer_2_reviewers=df[df.beer_name==beer_2].review_profilename.unique()
         common_reviewers=set(beer_1_reviewers).intersection(beer_2_reviewers)
```

0.1 Beer Recomendation system:

0.1.1 If there is a beer you like and want some recomnedation for a style of beer to try, then the system will do it for you. How does it work? There are three types of beer A,B,C. If Person 1 likes beer A and C, Person 2 likes beer A and C and You like beer C then:

0.1.2 You will also like beer A and you should be recommeneded beer A. In oredr to understand which beers are simlilar we are going to use "K nearest neighbors" approach.

```
In [23]: (df[df.beer_name==beer_1].review_profilename).head()

Out[23]: 1442361           gskitt
         1444203    savannahbrooks
         1444401       mtstatebeer
         1449783       Monkeyknife
         1450412              dawg
         Name: review_profilename, dtype: object
```

0.1.3 Get unique beer_names in the data

```
In [24]: df.beer_name.unique()

Out[24]: array(['Sausa Weizen', 'Red Moon', 'Black Horse Black Beer', ...,
                'Baron Von Weizen', 'Resolution #2', "The Horseman's Ale"], dtype=object)
```

0.1.4 Get the beer reviews for common users for a beer

```
In [25]: def get_beer_reviewrs(beer, common_users):
             mask=(df.review_profilename.isin(common_users))&(df.beer_name==beer)
             reviews=df[mask].sort('review_profilename')
             reviews=reviews[reviews.review_profilename.duplicated()==False]
             return reviews
```

In [31]:
```
common_users=common_reviewers
beer=beer_1
mask=(df.review_profilename.isin(common_users)) & (df.beer_name==beer)
```

In [32]:
```
reviews=df[mask].sort('review_profilename')
```

In [52]:
```
reviews=reviews[reviews.review_profilename.duplicated()==False]
```

In [74]:
```
 (get_beer_reviewrs(beer_1,common_reviewers)).head()
```

Out[74]:

	brewery_id	brewery_name	review_time	review_overall	review_aroma	review_appearance	review_profilename	beer_style	review_palate	review_taste	beer_name	beer_abv	beer_beerid
1454568	2681	Oskar Blues Grill & Brew	1221154773	3.5	4.0	3.5	ATPete	American Pale Ale (APA)	3.5	3.5	Dale's Pale Ale	6.5	6518
1453868	2681	Oskar Blues Grill & Brew	1297654968	4.5	4.0	4.5	AdamBear	American Pale Ale (APA)	4.0	4.5	Dale's Pale Ale	6.5	6518
1453766	2681	Oskar Blues Grill & Brew	1307229120	4.5	4.0	4.0	AlCaponeJunior	American Pale Ale (APA)	4.0	4.5	Dale's Pale Ale	6.5	6518
1454697	2681	Oskar Blues Grill & Brew	1210781469	4.5	3.5	4.5	AltBock	American Pale Ale (APA)	4.0	4.0	Dale's Pale Ale	6.5	6518
1454131	2681	Oskar Blues Grill & Brew	1267989965	4.0	4.0	4.0	Andreji	American Pale Ale (APA)	4.0	4.0	Dale's Pale Ale	6.5	6518

In [38]:
```
beer_1_reviews= get_beer_reviewrs(beer_1,common_reviewers)
beer_2_reviews= get_beer_reviewrs(beer_2,common_reviewers)
ALL_FEATURES=['review_overall','review_aroma','review_palate','review_taste']
```

In [39]:
```
cols=['beer_name','review_profilename','review_overall','review_aroma','review_palate','review_
beer_2_reviews[cols].head()
```

Out[39]:

	beer_name	review_profilename	review_overall	review_aroma	review_palate	review_taste
202456	Fat Tire Amber Ale	ATPete	4.5	4.0	4.0	4.5
201458	Fat Tire Amber Ale	AdamBear	3.5	2.5	4.5	3.5
201886	Fat Tire Amber Ale	AlCaponeJunior	2.0	3.0	3.5	3.0
202481	Fat Tire Amber Ale	AltBock	4.0	3.0		
201803	Fat Tire Amber Ale	Andreji	4.0	4.5		

```
         202481            3.0           3.0
         201803            4.0           4.0
```

```
In [40]: from sklearn.metrics.pairwise import euclidean_distances
```

```
In [41]: def calculate_similarity(beer1,beer2):
             #find common reviewrs
             beer_1_reviewers=df[df.beer_name==beer1].review_profilename.unique()
             beer_2_reviewers=df[df.beer_name==beer2].review_profilename.unique()

             #find users who reviewed beer1 and beer2
             common_reviewers=set(beer_1_reviewers).intersection(beer_2_reviewers)

             #get reviews for beer1 and beer2
             beer_1_reviwers=get_beer_reviewrs(beer1,common_reviewers)
             beer_2_reviwers=get_beer_reviewrs(beer2,common_reviewers)
             dists=[]
             #find the euclidean distances between beer1 and beer2
             for f in ALL_FEATURES:
                 dists.append(euclidean_distances(beer_1_reviwers[f],beer_2_reviwers[f])[0][0])
             return dists
```

```
In [42]: f=ALL_FEATURES[2]
         print(f)
         euclidean_distances(beer_1_reviews[f],beer_2_reviews[f])[0][0]
```

```
review_palate
```

```
Out[42]: 16.522711641858304
```

```
In [309]: euclidean_distances(beer_1_reviews[f],beer_2_reviews[f])
```

```
Out[309]: array([[ 16.52271164]])
```

```
In [310]:     beer_1_reviwers=df[df.beer_name==beer_1].review_profilename.unique()
              beer_2_reviwers=df[df.beer_name==beer_2].review_profilename.unique()
              common_reviewers=set(beer_1_reviwers).intersection(beer_2_reviwers)
```

```
In [311]: beer_1_reviwers
```

```
Out[311]: array(['gskitt', 'savannahbrooks', 'mtstatebeer', ..., 'krausenman',
                 'MrDonQuixote', 'jondeelee'], dtype=object)
```

```
In [312]: common_reviewers=set(beer_1_reviwers).intersection(beer_2_reviwers)
```

```
In [313]: calculate_similarity(beer_1,beer_2)
```

```
Out[313]: [17.656443583009576,
           17.449928366615147,
           16.522711641858304,
           17.599715906798043]
```

```
In [ ]:
```

```
In [315]: list1.append(1)
          list1
```

```
Out[315]: [1]
```

```
In [316]: list1.append(2)
          list1

Out[316]: [1, 2]

In [44]: beers_list=df.beer_name.unique()
         beers_list

Out[44]: array(['Sausa Weizen', 'Red Moon', 'Black Horse Black Beer', ...,
                'Baron Von Weizen', 'Resolution #2', "The Horseman's Ale"], dtype=object)

In [45]: beers=["Dale's Pale Ale","Sierra Nevada Pale Ale","Michelob Ultra",
                "Natural Light","Bud Light","Fat Tire Amber Ale","Coors Light",
                "Blue Moon Belgian White","60 Minute IPA","Guinness Draught"]

In [46]: def task1(beer1,beers):
             #find common reviewrs
             empty_list=[]
             for i in range(len(beers)):
                 #print(beers[i])
                 #print(calculate_similarity(beer1,beers[i]))
                 empty_list.append(calculate_similarity(beer1,beers[i]))
             print empty_list

In [ ]:

In [47]: beers[1],beers[0],beers[7]

Out[47]: ('Sierra Nevada Pale Ale', "Dale's Pale Ale", 'Blue Moon Belgian White')

In [48]: calculate_similarity(beer_1,beers[8])

Out[48]: [17.832554500127006, 17.38533865071371, 16.568041525780892, 16.61324772583615]

In [49]: calculate_similarity("Sausa Weizen","Red Moon")

Out[49]: [1.5, 0.5, 1.5, 1.5]

In [52]: task1(beers[0],beers)

[[0.0, 0.0, 0.0, 0.0], [18.553975315279473, 15.771810295587505, 15.748015748023622, 16.201851746019649]
```

0.2 Find all euclidean distances for earch of beer pairs in beers list:

```
In [61]: def task2(beers):
             #find common reviewrs
             simple_list=[]
             for beer1 in beers:
                 #print(beer1)
                 for beer2 in beers:
                     #print(beer1+'-'+beer2)
                     if beer1!=beer2:
                         row=[beer1,beer2]+calculate_similarity(beer1,beer2)
                         #print(row)
                         #print("")
                         simple_list.append(row)
```

```
    return(simple_list)
simple_distances=task2(beers)
simple_distances
cols=["beer1","beer2","overall_dist","aroma_dist","palate_dist","taste_dist"]
simple_distances=pd.DataFrame(simple_distances,columns=cols)
simple_distances.head()
```

Out[61]:

	beer1	beer2	overall_dist	aroma_dist	palate_dist	taste_dist
0	Dale's Pale Ale	Sierra Nevada Pale Ale	18.553975	15.771810	15.748016	16.201852
1	Dale's Pale Ale	Michelob Ultra	28.626910	30.504098	29.841247	31.519835
2	Dale's Pale Ale	Natural Light	23.021729	25.865034	23.606143	26.186829
3	Dale's Pale Ale	Bud Light	38.147739	40.574006	38.298172	41.590263
4	Dale's Pale Ale	Fat Tire Amber Ale	17.656444	17.449928	16.522712	16.599716

0.2.1 Weights are user inputs. As a person asking for recommendation, if I prefer to value taste over aroma, I will give higher weight to taste (less value of weight means more importance). This way the recommendation will be biased towards taste more than aroma.

In [60]:
```
def calc_distance(dist,beer1,beer2,weights):
    mask=(dists.beer1==beer1)&(dists.beer2==beer2)
    row=dists[mask]
    row=row[['overall_dist','aroma_dist','palate_dist','taste_dist']]
    dist=weights*row
    return dist.sum(axis=1).tolist()[0]
```

In [62]:
```
weights=[2,1,1,1]
dists=simple_distances
beer1=beer_1
beer2=beer_2
```

In [63]:
```
mask=(dists.beer1==beer1)&(dists.beer2==beer2)
row=dists[mask]
row=row[['overall_dist','aroma_dist','palate_dist','taste_dist']]
dist=weights*row
row.head()
```

Out[63]:

	overall_dist	aroma_dist	palate_dist	taste_dist
4	17.656444	17.449928	16.522712	17.599716

0.2.2 The final weighted values of euclidean distances calculated for each pair of the beer

In [64]:
```
dist.head()
```

Out[64]:

	overall_dist	aroma_dist	palate_dist	taste_dist
4	35.312887	17.449928	16.522712	17.599716

In [65]:
```
dist.sum(axis=1).tolist()[0]
```

Out[65]: 86.885243081290639

```
In [66]: calc_distance(simple_distances,"Fat Tire Amber Ale","Dale's Pale Ale",weights)

Out[66]: 86.885243081290639

In [67]: calc_distance(simple_distances,"Fat Tire Amber Ale","Michelob Ultra",weights)

Out[67]: 153.00571860654327
```

0.2.3 Let's make prediction for person having "Coors Light" beer

```
In [68]: my_beer="Coors Light"
         result=[]
         for b in beers:
             if my_beer!=b:
                 result.append((my_beer,b,calc_distance(simple_distances,my_beer,b,weights)))
```

0.2.4 The sorted list of sorted beers with clalculated euclidean distances

```
In [70]: sorted(result,key=lambda x:x[2])

Out[70]: [('Coors Light', 'Natural Light', 70.483724369544859),
          ('Coors Light', 'Michelob Ultra', 71.312866260028784),
          ('Coors Light', 'Bud Light', 101.35815659584495),
          ('Coors Light', 'Blue Moon Belgian White', 174.68407232789718),
          ('Coors Light', 'Fat Tire Amber Ale', 175.74577705697465),
          ('Coors Light', "Dale's Pale Ale", 180.69937116048874),
          ('Coors Light', 'Guinness Draught', 204.99494753154124),
          ('Coors Light', '60 Minute IPA', 233.68501559769081),
          ('Coors Light', 'Sierra Nevada Pale Ale', 255.00673514359349)]
```

0.2.5 The "Natural Light" beer should be recomended for a person who liked "Coors Light", based on the histirical data.

```
In [73]: sorted(result,key=lambda x:x[2])[0][1]

Out[73]: 'Natural Light'

In [ ]:
```