Data Science IBM Specialization

Coursera

CITIES CLASSIFICATION

Analysis And Clustering
Of The Cities Around The World

Lucas Daniel Tomasini

November 21, 2018

Content Table

Introduction – Business Problem

The problem that we are interested in here is the clustering of different cities around the world that share similar characteristics. More specifically, it is about categorizing cities according to different types of tourist activities that can be carried out there. For example, there may be cities full of historic sites, cities surrounded by natural landscapes (mountains, rivers), cities with adventure tourism activities (mountaineering, skiing, snorkeling), cities with a lot of musical culture, cities with plenty of paradisiacal beaches or cities with great nightlife.

This study is meant to help tourism agencies to build not only standard travel packs, but also especially custom-made packages adapted to the needs and requirements of the customer.

Data Set and Cleaning

For this study, a sample of 53 cities with varied characteristics was selected. The first step was to find the most relevant venues belonging to a given category in every city. To that end the Foursquare API was used with the explore functionality. One request to that end-point was run per city and per category, with the following parameters:

> *near*: a string naming a place in the world (e.g. Berlin, Germany). We used this parameter instead of ll (latitude, longitude)
>
> *query*: a term to be searched against a venue's tips, category, etc. For example, "Historic Site", or "Beach". Here we specify the category of the venues we want to search.
>
> *time = "any"* & *day ="'any"*: we passed "any" to retrieve results for any time of day and any day of the week. Omitting these parameters returns results targeted to the current time of day or the current day of week. And we want this analysis not to be time dependent.

We did not specify a radius, so as a suggested one be used based on the density of venues in the area, because we do not know in advance how large the city and its surroundings can be.

After doing a little research on the existing categories in the Foursquare platform, we came up with the following 25 list items for this study:

> Palace – Church – History Museum – Art Museum – Art Gallery – Theater – Restaurant – Dancing – Beer Bar – Wine Bar – Cocktail – Trail – Mountain – Lake – Beach – Park – Concert Hall – Music & Shows – Stadium – Ski Area – National Park – River – Gourmet – Historic Site – Snorkeling

We executed a total of 53 x 25 = 1325 requests to API to get the most relevant venues per city per category. We stored all this information into a data frame, getting a total of 19825 venues, as shown below:

```
In [8]: df_venues.shape
Out[8]: (19825, 9)
```

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

Out[4]:

	venue_id	name	category	lat	long	keyword	city	country	city_location
0	4adcda09f964a520df3321e3	Château de Versailles	Palace	48.804729	2.120340	Palace	Paris	France	48.85341, 2.3488
1	4bc08f79461576b014487a32	Petit Trianon	Palace	48.815222	2.109872	Palace	Paris	France	48.85341, 2.3488
2	4be58f71cf200f474d57133c	Grand Trianon	Palace	48.814213	2.104874	Palace	Paris	France	48.85341, 2.3488
3	4adcda10f964a520af3521e3	Musée du Louvre	Art Museum	48.860847	2.336440	Palace	Paris	France	48.85341, 2.3488
4	4c9f3d1b46978cfaee0ea97f	Salon de Mercure	Historic Site	48.805153	2.120654	Palace	Paris	France	48.85341, 2.3488

As you can see from this table, the keyword column is the query term used for the search and the category column is the actual category of the venue as it appears in Foursquare. We want to keep only the venues whose categories match or be related to the search keyword (e.g. we do not want a historic site when asked for a palace, because historic sites will be collected with another keyword: “Historic Site”). After this filtering we keep 13150 venues:

```
In [12]: df_venues.shape
Out[12]: (13150, 9)
```

```
In [15]: df_venues.head()
```

Out[15]:

	venue_id	name	category	lat	long	keyword	city	country	city_location
0	4adcda09f964a520df3321e3	Château de Versailles	Palace	48.804729	2.120340	Palace	Paris	France	48.85341, 2.3488
1	4bc08f79461576b014487a32	Petit Trianon	Palace	48.815222	2.109872	Palace	Paris	France	48.85341, 2.3488
2	4be58f71cf200f474d57133c	Grand Trianon	Palace	48.814213	2.104874	Palace	Paris	France	48.85341, 2.3488
15	4cb5f18de262b60cb95a6be0	Château de Guermantes	Palace	48.852891	2.697893	Palace	Paris	France	48.85341, 2.3488
20	4adcda09f964a520e83321e3	Cathédrale Notre-Dame de Paris	Church	48.853124	2.349561	Church	Paris	France	48.85341, 2.3488

We need to be able to compare the importance of the different categories between each city. That is why we need a set of parameters that can help us determine how important a venue of a given category is compared to others from the same category. The Foursquare API has an endpoint named *venues* that gives us the full details about a venue including rating, likes, tips, etc. The second step in the data collection is to obtain this information for every venue from the previous step. One request per venue was made to *venues* endpoint with the following parameters:

venue_id: ID of the venue to retrieve

The information extracted from the response was:

rating: numerical rating of the venue (0 through 10). Not all venues will have a rating.

ratingSignals: The count of users who have rated this this venue

likes: The count of users who have liked this venue

tipCount: Contains the total count of tips.

This is a premium request, therefore several days and five different accounts were needed to execute all of them. The collected data was store in a second data frame:

```
In [26]: df_venues_scores.shape
Out[26]: (13150, 9)
```

```
In [24]: df_venues_scores.head()
```

Out[24]:

venue_id	rating	rating_count	tip_count	likes_count
4adcda09f964a520df3321e3	9.2	4484.0	753.0	3686.0
4bc08f79461576b014487a32	9.0	281.0	12.0	250.0
4be58f71cf200f474d57133c	8.8	303.0	14.0	260.0
4cb5f18de262b60cb95a6be0	7.6	17.0	2.0	15.0
4adcda09f964a520e83321e3	9.6	9207.0	962.0	8165.0

The third step was to join the two data frames into one, getting a complete list of venues with the details. Then we removed all the venues which had not a rating:

```
In [30]: df_venues_detail.head()
```

Out[30]:

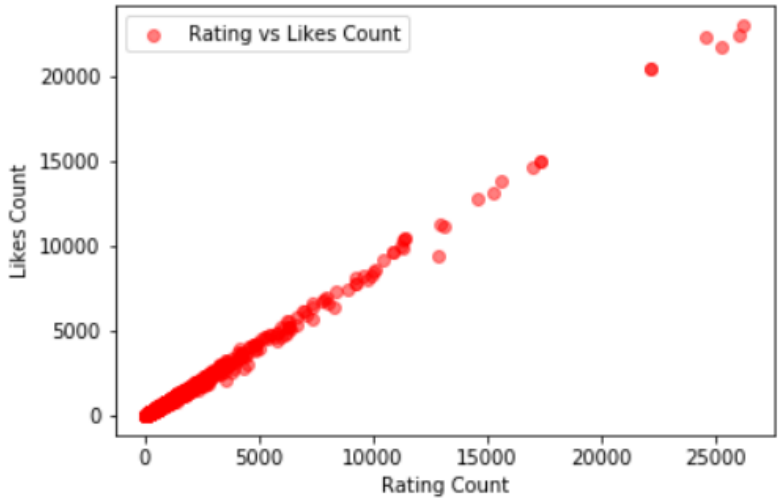
	venue_id	name	category	lat	long	keyword	city	country	city_location	rating	rating_count	tip_count	likes_count
0	4adcda09f964a520df3321e3	Château de Versailles	Palace	48.804729	2.120340	Palace	Paris	France	48.85341, 2.3488	9.2	4484.0	753.0	3686.0
1	4bc08f79461576b014487a32	Petit Trianon	Palace	48.815222	2.109872	Palace	Paris	France	48.85341, 2.3488	9.0	281.0	12.0	250.0
2	4be58f71cf200f474d57133c	Grand Trianon	Palace	48.814213	2.104874	Palace	Paris	France	48.85341, 2.3488	8.8	303.0	14.0	260.0
15	4cb5f18de262b60cb95a6be0	Château de Guermantes	Palace	48.852891	2.697893	Palace	Paris	France	48.85341, 2.3488	7.6	17.0	2.0	15.0
20	4adcda09f964a520e83321e3	Cathédrale Notre-Dame de Paris	Church	48.853124	2.349561	Church	Paris	France	48.85341, 2.3488	9.6	9207.0	962.0	8165.0

The *df_venues_detail* data frame will be used as the input data for the next section, where we will analyze and transform this data in a way that best suits the subsequent analysis.

Methodology

We will apply the k-means algorithm in order to group cities according to similar categories. The input features for the algorithm will be a matrix (with rows for cities and columns for categories) containing a unique score for each category of a given city. In this case, we have 53 cities and 25 categories, so the matrix will be 53 x 25.

In order to find a unique number that adequately represents the importance of a category in a city, we have to analyze the input variables information obtained in the *df_venues_detail* data frame: rating, rating count, tip count and likes count. Let's first look at the relationship between rating count and likes count. A scatter plot between the two variables and their correlation coefficient are shown below:

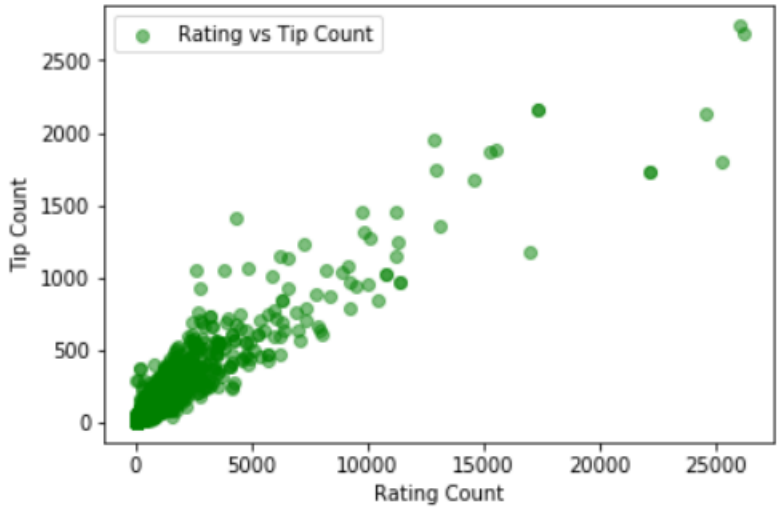



```
In [34]: df_venues_detail['likes_count'].corr(df_venues_detail['rating_count'])
```

```
Out[34]: 0.9981254853147993 9
```

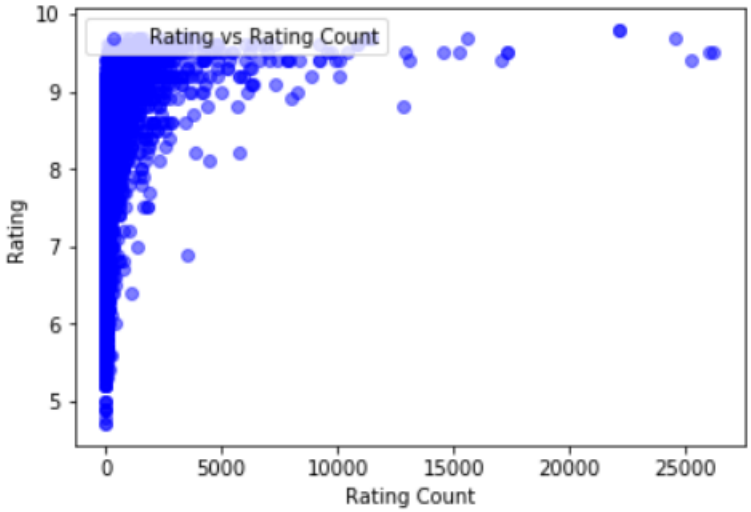
Both the rating count and the likes count are highly correlated (99.8% of correlation). That means that we can perfectly use only one of them, since they both provide the same information. We will keep rating counts and get rid of likes count.

Now we will do the same for tip count vs rating count and rating vs rating count:




```
In [36]: df_venues_detail['tip_count'].corr(df_venues_detail['rating_count'])
```

```
Out[36]: 0.93246485206500362
```




```
In [38]: df_venues_detail['rating'].corr(df_venues_detail['rating_count'])
```

```
Out[38]: 0.26667196713801883
```

As we can see, rating count and tip count are also highly correlated (93.2% of correlation), so we will take a similar approach and drop de tip count. Finally, rating and rating count are weakly correlated, meaning that both variable provide different kind of information. In fact, two venues could have the same high rating, and yet one of them be rated by a larger number of people. So the reputation of a venue depends on both the rating and the number of people that rated it.

From the previous analysis, a final score for each city and category will be computed as follow: let $r_{i,j}^k$ and $c_{i,j}^k$ be the rating and rating count, respectively, of the *k* venue that belongs to the *i-th* city and the *j-th* category. The final score $s_{i,j}$ for city *i* and category *j* is equal to:

$$s_{i,j} = \sum_k (r_{i,j}^k \,.\, c_{i,j}^k)$$

The following data frame shows the computed scores for each city/category (the image is truncated due to its size):

	country	city	Ski Area	Restaurant	Mountain	Trail	Dancing	Beach	National Park	Beer Bar	History Museum	Theater	Cocktail	Park	Concert Hall	Historic site	Stadium	Art Museum	Art Gallery	Wine Bar	Music & Shows
43	South Korea	Seoul	0.0	16319.5	4027.3	9218.9	6818.2	0.0	916.1	10970.3	25337.2	7472.6	10847.1	44268.3	12438	32282.5	20930.7	34232.8	10688.7	5509.8	808
44	Spain	Barcelona	0.0	108652.6	16280.1	7122.3	49002.6	90567.9	1101.1	40006.7	23907.8	27034.8	78289.6	127448.2	77783	134356	125577	98141.3	26566.5	40761.7	4392
45	Spain	Granada	0.0	21185.4	85.8	212.5	4902.5	0.0	0.0	623.1	3166.5	508.0	2062.3	3627.7	1033.2	38988.4	1623.5	1673.5	486.3	9503.2	1857.6
46	Taiwan	Taipei	0.0	33887.4	3905.7	5935.4	4745.6	0.0	2145.5	3510.3	11959.8	3242.0	12550.9	17183.8	1929.1	8931.2	3297.7	5590.2	7546.1	2099.5	1951.7
47	Thailand	Bangkok	0.0	38135.7	0.0	0.0	46199.8	603.1	0.0	9724.5	14291.3	10773.8	36428.3	58634.9	8669.2	31231.9	17535.2	26662.6	29717.9	22993.2	179.2
48	Turkey	Istambul	1254.1	71012.1	33198.6	51574.8	279010.0	127262.2	5695.7	169012.7	322493.5	134503.7	221874.3	313898.7	252161	956187	638383	73869.1	70020.2	146464	0
49	United Kingdom	London	915.2	128013.3	0.0	8672.1	48516.9	404.0	0.0	63987.5	216692.1	91380.8	95014.5	315021.1	81291.7	120601	96282.5	238890	86533.6	35732.2	67760.6
50	United States	Los Angeles	0.0	65529.6	2449.9	41354.7	42453.8	116928.9	0.0	24863.0	27135.0	68000.8	51728.3	30570.5	102432	46811.9	91413.1	93118.7	13677.3	45401.4	47531.4
51	United States	New York	180.8	120556.4	0.0	35629.6	115253.9	79.2	1616.0	108589.8	180365.2	125928.7	175136.3	711435.5	117648	203078	94427	459050	29678.4	72152.5	12429.4
52	Vietnam	Hanoi	0.0	12956.9	54.0	0.0	4109.9	0.0	0.0	1872.9	5835.2	2558.3	3522.0	2309.3	1577.9	10960.9	442.2	608.4	1650.5	4739.2	490.3

```
: df_cities_info.shape
```

```
7]: (53, 27)
```

This data frame will be the input to the k-mean algorithm. We will present the results in the next section.

Results

We will apply the k-means algorithm in order to group cities according to similar categories. We chose 6 as the number of clusters, following the elbow procedure for getting the optimum. The results are:

```
]: df_cities_info.sort_values(by = 'labels')[['country', 'city', 'labels']]
```

56]:

	country	city	labels
19	Egypt	Cairo	0
34	Myanmar	Bagan	0
32	Mexico	Guadalajara	0
43	South Korea	Seoul	0
44	Spain	Barcelona	0
28	Italy	Rome	0
27	Italy	Florence	0
25	India	New Delhi	0
23	Hungary	Budapest	0
21	Germany	Berlin	0
20	France	Paris	0
18	Czech Republic	Prague	0
39	Peru	Lima	0
6	Austria	Vienna	0
47	Thailand	Bangkok	0
10	Canada	Toronto	0
48	Turkey	Istambul	0
49	United Kingdom	London	0
46	Taiwan	Taipei	0

13	Chile	Valparaiso	1
4	Australia	Melbourne	1
29	Japan	Tokyo	1
5	Australia	Sydney	1
51	United States	New York	1
22	Hong Kong	Hong Kong	1
1	Argentina	Buenos Aires	1
40	Philippines	Manila	1
36	Netherlands	Amsterdam	1
15	China	Shanghai	1
14	China	Pekin	1
24	India	Mumbai	1
42	South Africa	Cape Town	1
41	Russia	Sochi	2
7	Brazil	Manaus	2
8	Brazil	Recife	2
17	Colombia	Santa Marta	2
16	Colombia	Cartagena	2
52	Vietnam	Hanoi	2
50	United States	Los Angeles	3
26	Indonesia	Bali	3
37	New Caledonia	Noumea	3
31	Malaysia	Borneo	3
12	Chile	Isla de Pascua	3
11	Chile	Iquique	3
9	Brazil	Rio de Janeiro	3
45	Spain	Granada	4
3	Argentina	Ushuaia	4
2	Argentina	Mendoza	4
38	Peru	Arequipa	4
0	Argentina	Bariloche	4
35	Nepal	Kathmandu	5
33	Morocco	Marrakech	5
30	Kenya	Nairobi	5

Discussion

As we can see from the clustering result:

- The first group (label = 0) corresponds mostly to old cities with historic places and vast culture (e.g. Rome, Florence, Vienna, El Cairo, Paris, Berlin)
- Label 1 may belong cities with an active nightlife, such as: Buenos Aires, Valparaíso, New York, Amsterdam
- For label 2 we have small colonial cities: Cartagena, Santa Marta
- For label 3 there are mostly islands, cities with beaches and water activities.
- Label 4 are quieter cities surrounded by mountains and with trekking activities.
- Label 5: another category that should be further analyzed.

Conclusion

Although this study was carried out using only a few cities, it could be extended to thousands of cities around the world, allowing to group them in clusters of similar tourist characteristics. This will help agencies build customs packages to its customers in an easy and automated way.