AirBnB: Data Analysis using Spark

Thomas Kleiven og Karsten Standnes

March 2017

1 Introduction

This assignment consists of analysing datasets containing information regarding listings at AirBnB. In order to analyse the data, we have used the Apache Spark framework a long side with python.

AirBnB is an online hospitality service, that allow people to list or rent short-term lodging like holiday rentals, aparment rentals, hotel rooms, etc. The company itself does not own any lodging, but receives commissions from both guests and hosts in conjunction with each booking. It has more than 3.000.000 lodgning listings in 65.000 cities in 191 countries. The dataset we have been given focuses on three cities, namly Seattle, San Fransisco and New York. Furthermore, the data sets contains information about each listing such as when it was/have been available for rental, customer reviews, etc.

Figure 1: Spark logo

Contents

2 Exploration of the Data Set

It is stated in the assignment that one should find appropriate Spark functions in order to solve each subproblem. To ensure easy accessing we have decided to use pyspark's SQL functions. We turn each file into a dataframe, and then perform simple SQL-quieries in order to produce the results.

To get the information into dataframes we first have to use RDD-functions such as map in order to split the tab-seperated lines. Secondly, we then use filter to filter the header and lastly we use map again to assign each column to the dataframe.

3 Task 2

1. In the dataset each file has a primary key called 'id'. To be able to join the different tables together both 'reviews' and 'calender' has foreign keys to 'lisitings' which is the main table in this set.

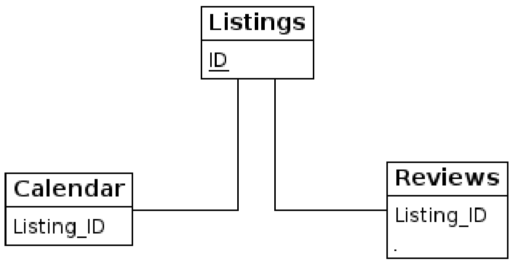

Figure 2: Visualization of foreign keys

2. Number of distinct values in each column in listings is given by iterating through all the columns and applying the distinct- and count-functions on the dataframe. The results are too big for this report and can therefore be seen in the attachments.

3. It is three cities in total in the data set, these are New York, San Fransisco and Seattle.

4. In this assignment we find these five fields in the listings dataset important:

   - host_id - Uniquie identifier of each host, most likely to find integer values in this column.
   - host_name - It is interesting to know the hosts name. Even though a hosts name is not necessarily unique, we find this field important.
   - Price for one night - Most likely to contain integer values from $50 to $2000
   - Price for enitre house and price for private room - It is nice to know whether a private room is much cheaper than entire house. Most likely to contain integer values.
   - Accomodations - It is helpful to know how many nights each listing has been booked in the past, most likely to contain integer values.

4 Task 3

(a) Select wanted attributes from the listings dataframe such as AVG(price) and city.

City	Average booking price pr. night
New York	\$145
San Fransico	\$250
Seattle	\$131

Figure 3: Table showing the average booking price pr. night in each city

(b) Select city, roomtype and AVG(price) from the listings dataframe

City	Entire home/apartment	Private room	Shared room
New York	\$207	\$90	\$69
San Fransisco	\$341	\$135	\$88
Seattle	\$160	\$77	\$51

Figure 4: Table showing the price pr. room in each city

(c) Select city and AVG(reviews_per_month) from listings

City	Number of reviews pr. month
New York	53.890
San Fransisco	11.463
Seattle	6.633

Figure 5: Table showing the number of reviews pr. month in each city

(d) Estimated number of nights booked per year is given by summing the reviews_per_month column. We then multiply with 3 because each guest on average stays three nights and then we divide by 0.7 because about 70% of the guests leave a review. We also multiply by 12 in order to get number of nights booked pr. year. We then get the result

City	Estimated number of nights booked pr. year
New York	2.771.470
San Fransisco	589.537
Seattle	341.168

Figure 6: Table showing the estimated number of nights booked pr. year in each city

(e) Estimated amount of money spent in dollars are given by summing the average price pr. room for those dates where the rooms have been booked, we then get

City	Estimated amount of money spend pr. year in $
New York	412.727.275
San Fransisco	147.820.512
Seattle	44.703.295

Figure 7: Table showing the estimated amount of money spent

5 Task 4

(a) In order to find the global average of listings pr. host we choose host_total_listing_count from listings and compute the average.

$$Global\ average\ of\ listings\ pr.\ host = 3.3 \tag{1}$$

(b) The percentage of hosts with more than one listing is easily calculated by filtering the hosts from host_total_listing_count in the listings dataset that have more than one listing, then we perform

$$\frac{Number\_of\_hosts\_with\_more\_than\_one\_listing}{Number\_of\_hosts\_with\_at\_least\_one\_listing} = 0.18 \tag{2}$$

(c) In order to find top three hosts in each city that has the highest outcome, we have to join our listing set with the calendar set because we need to figure out when each listing was booked. Let us join on

$$listing.id = calendar.listing\_id \tag{3}$$

From this we choose city, host_id, host_name, and we accumulate the price for each of the hosts listings. We then partition over the cities to select the three hosts in each city that has the highest income. We then arrive with the following result

City	Host Name	Host ID	Income in $
New York	Helena	59900772	5.721.125
	Olson	1235070	4.859.514
	Amy	3906464	4.069.593
Seattle	Daniela	8534462	2.423.533
	Darik	74305	1.660.563
	Sea To Sky Rentals	430709	1.478.297
San Fransisco	Suzy	18582461	4.147.500
	FirstName	45317952	3.950.000
	Kirk	50611045	3.950.000
	Jessica	8662557	3.949.605

Figure 8: Table showing top three hosts in each city ranked by their income

We will arrive with this result if, and only if, a host with a given ID only has one name.

6 Task 5

(a) The way to compute top three guests in each city is similair to the way in task 4c). We have to count how many reviews each guest has, assuming that a guest writes one review pr. booking. Let us join the listing table and the table containing the reviews on

$$listings.id = reviews.listing\_id \tag{4}$$

From this result we choose city, reviewer_id, reviewer_name and number of bookings. Then we partition over the cities selecting the top three guests with the highest number of bookings. We then arrive with

City	Host Name	Host ID	Number of Bookings
New York	J.B	197711	79
	Andy	2539165	41
	Adrienne	4236708	33
Seattle	Amanda	206203	71
	Kathryn	15121499	34
	David	5775807	27
San Fransisco	Emily	2895282	29
	Zafar	45106320	28
	Claire	18099543	25

Figure 9: Table showing top three guests in each city ranked by the number of bookings

(b) In order to find the guest who has spent the most money on accomomondations we have to join all of our three dataframes. Let us join on the following

$$listing.id = reviews.listing\_id \tag{5}$$

$$listing.id = calendar.listing\_id \tag{6}$$

We then accumulate the price of the places where each guest(reviewer) has rented and put the results in descending order. Then we see that Unnur Svava, with ID-number 1471097, has spent a total of $945000.

7 Task 6

(a) In order to assign a neighborhood to each listing we have decided to go with a slightly modified version of the **ray casting**-technique. In general, we run a semi-infinite ray horizontally with increasing x-value and fixed y-value out from our coordinate, and count how many edges it crosses. At each crossing, the ray has to switch between inside and outside of our polygon. This idea is inspired by the *Jordan curve theorem* that asserts that every *plane simple closed curve* divides the plane into an 'interior' region bounded by the curve and an 'exterior' region containing all of the nearby and far away exterior points, so that any *continous path* connecting a point of one region to a point of the other intersects with that loop somehere. [2]

Let us go through the algorithm in greater detail. We choose our listing with associated coordinate, and then itearate through all features/neighborhoods contained in the given .geojson consisting of the definitions of neighborhoods. In order to decide if our coordinate is inside a neighborhood (polygon) we first check if our coordinate's y-value is within each of the edge's scope in our polygon. Secondly, we check whether our coordinate is to the left of our polygon's edges. We do this by comparing the slope of the line drawn from our coordinate to the polygon's corners. We do this for all of the edges in our polygon. The test will look like this [3]

$$\Delta x_p < \frac{\Delta x \cdot \Delta y_p}{\Delta y} \tag{7}$$

where Δx_p is the change in x-value from our coordinate to the edge's starting point, Δx is the edge's change in x-value, Δy is the change in the edge's y-value and Δy_p is the change in y-value from our point to the edge's starting point.

If the result of this test is true, the line drawn rightwards from our coordinate-point, by definition, has to cross that edge. By repeatedly counting hos many times the rightward line crosses the polygon we can decide whether our point is on the inside or outside. If it is an even number, we know that the point is on the outside, likewise, if it is odd our coordinate has to be within this neighborhood.

This approach works quite well in our opinion. In New York we get that 80% of the listings is in the correct neighborhood when plotted in carto.com together with the .geojson-file. In San Francisco 95% of the listings are in the correct neighborhood, and in Seattle 95% of the listings the correct. The calculations for these results can be seen in 'neighbor.py'. These queries found in 'neighbor.py' is to be performed at Carto, using the SQL-framwork alongside with the attached neighborhood files for the three cities. The provided test-file only has a percentage hit of 57% when it comes to which listing is in the correct neighborhood. When we compare our results directly with the test-file, about 23% are equal. Numerical percision will affect our accuracy score and this, in addition to inaccuracy in the provided test file, is the main reasons why our data does not match the test-results too well.

(b) In order to find the distinct set of amenities belonging to the listings within the neighborhood, we join our results from assignment 6a) with the listings dataframe. We have written our results to file in task 6a). These results are made into dataframe and then joined on the ID key in the listings dataframe in order to find the distinct amenities for each neighborhood. We then count up the distinct amenities for each neighborhood. The results for this assignment can be seen in 'task6b.csv' provided in the attachment.

8 Visualization with Carto

To visualize the data found in the previous tasks we used Carto. As mentioned in Task 6 we used Carto to also check how well the algorithm had performed in placing the geographical points. We measured this performance by checking how high a percentage of the points had the right neightbourhood value using the sql functionality on Carto. This is quite simple with the compatability that the sql function on Carto has with PostGIS. To find how many are within the correct neighbourhodd we use the function S_intersects[4]. In this function takes the the_geom columns from the tabel with the points we obtained from the algortihm and the .geojson file with neighbourhoods. This gave the results referred to above.

The maps have been visualized by taking different attributes from listings and joining them with the results. In Seattle there is applied a choropleth map which give a visual on the price for each facility. In New York and San Francisco we made choropleth maps that takes in the review rating for each listing. In San Francisco it is also possible to click on each point and see a picture of the appartment. All the code for obtaining the csv-files uploaded to Carto can be found in 'neighbor.py'. For pratical reasons of size we assume that the csv-files 'listings$_u s.csv','reviews_u s.csv'and'calender_u s'areavailable.$

Under are hyperlinks to all the cities:

- Seattle
- New York
- San Francisco

References

[1] Apache Spark logo http://spark.apache.org/images/spark-logo-trademark.png

[2] https://en.wikipedia.org/wiki/Jordan_curve_theorem

[3] http://stackoverflow.com/questions/217578/how-can-i-determine-whether-a-2d-point-is-within-a-polygon

[4] http://postgis.net/docs/ST_Intersects.html