Solution to data challenge by Susha Pozhampallan Suresh for Machine Learning Engineer in Coveo

Question 1: There was a bug in the code and for one specific country, the records don't have the country field logged. It just shows up as an empty field (""). The search sessions with a missing country either come from a country that is completely missing from the data, or from one of the countries that are logged in the data. Can you determine which country it is the most likely to be? Explain your hypothesis and the data analysis tasks you did to find the missing country.

Answer: Yes, it is possible to determine the missing country. I analysed the data of user country and the total number of searches from each hour. So I had a table with an hour and for each country(including the missing country) the total number of searches performed each hour. My hypothesis was that people usually only search during day time and this hypothesis was confirmed by the data. The search pattern of missing countries shows patterns similar to the European countries in the system. So my guess is the country with a time zone ±3 GMT. It is most likely Russia which is GMT +3.

The table is shown below:

session_hour	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Total
country																									
	278	293	289	270	282	282	276	163	85	40	13	4	1	0	0	0	0	0	0	1	33	87	168	255	2820
DE	29	75	151	278	391	394	328	357	341	385	332	272	169	72	34	13	4	0	0	0	0	0	3	10	3638
ES	20	45	95	158	196	218	203	156	190	230	167	134	77	40	13	5	0	1	0	0	0	0	0	5	1953
FR	12	57	117	164	213	235	222	226	241	233	216	182	106	43	17	9	3	0	0	0	0	0	1	1	2298
IT	22	36	94	142	177	201	165	180	182	196	167	148	94	42	22	5	0	0	0	0	0	0	4	5	1882
UK	20	32	77	162	273	321	375	343	319	375	368	369	238	177	72	20	6	1	1	0	0	0	0	6	3555
US	3	2	1	0	0	1	8	17	49	98	184	277	355	427	376	413	359	361	351	257	180	105	41	11	3876
Total	384	540	824	1174	1532	1652	1577	1442	1407	1557	1447	1386	1040	801	534	465	372	363	352	258	213	192	217	293	20022

Table 1: Analysis of search pattern on an hourly basis for users from different countries.

The codes are present in ***Assignment_missing_country.ipynb*** file.

Question 2: Given a sequence of searched cities, find the most likely city or cities to be also searched next, within the same session. Your code should include a function that takes a list of 0 to n cities and returns the most likely next city or cities. Keep in mind that the goal is to call this function each time a user performs a search. It should therefore be fast to execute.

Answer: I have tried market basket analysis, association rules based on apriori algorithm and fp tree algorithm. This method helps us analyse sequence of patterns and determine the association between the items that occur together. One other method that I have tried is analysing the similarity using the cosine similarity between the frequent searches.

Solution to data challenge by Susha Pozhampallan Suresh for Machine Learning Engineer in Coveo

Association rule mining using fp-tree growth and apriori: I followed the following steps in the solution using fp tree and apriori.
I Checked the search session to ensure if it has all unique values, no missing values and I preprocessed the json file. The program that I have has the following functions:

a. Load_data : that loads the data from json file to a dataframe for ease of analysis.
b. Sparse_transaction_encoder: which encodes the transaction data in form of a Python list of lists into a NumPy array.
c. Association_rule: Apriori function is a popular algorithm [1] to extract frequent itemsets for association rule mining, with a support of 0.001. A frequent itemset is defined as a set of items that occur together in at least 1% of all transactions in the data. Another popular alternative that I tried is FP-tree (frequent pattern tree) data structure. This algorithm is faster than the apriori algorithms in terms of execution time. Association rule is applied to the frequent item set to find the association between them. It is based on the rule {antecedent} →{consequent}. Given a list of searched cities using association rules, I found the next possible city to be searched.
d. Prediction_cities: This function compares the list of searched cities and returns the city with highest confidence value from the possible consequences

Compared to the apriori algorithm, the fp tree was faster in terms of execution speed. By changing the threshold values, we can further optimize the algorithm.

The code is in ***Assignment_Association_rule_mining.py.***

Association rule mining using similarity matrix: In this method, I calculated the column-wise cosine similarity of a sparse matrix of frequent items. The algorithm takes as input a city and returns an ordered list of 0-n cities that can be searched next based on their similarity scores. The cose is in ***Association_using_similarity-matrix.ipynb.***

There are few features describing each user: user id, joining date and country. Are these features useful to predict the most likely city to be searched? How do they compare to the other features tried in Question 2 (i.e. previous cities searched)? Can the algorithm implemented in Question 2 be improved by making use of these features?

Question 3: There are few features describing each user: user id, joining date and country. Are these features useful to predict the most likely city to be searched? How do they compare to the other features tried in Question 2 (i.e. previous cities searched)? Can the algorithm implemented in Question 2 be improved by making use of these features?

Solution to data challenge by Susha Pozhampallan Suresh for Machine Learning Engineer in Coveo

Answer: Yes these features can be used to predict the most likely city to be searched next, especially the country of the users. I tried to grow fp trees for specific countries and the confidence of prediction has improved compared to question 2 as I tried the second question with the cities searched. Below the prediction of the next city when the searched Chicago is New York with a confidence of 0.1413 ie 14.16%.

```
# the confidence of prediction of next city when the search is Chicago IL for fp-tree trained on US

city = {'Chicago IL'}
match = rule[rule['antecedents'] == city]
print("The next possible city to be searched:", match.loc[match['confidence'].idxmax()]['consequents'])
print("The confidence of predicscore:",match.loc[match['confidence'].idxmax()]['confidence'])
```

```
city: frozenset({'New York NY'})
score: 0.14138817480719795
```

Below the prediction of the next city when the searched Chicago previously is New York with a confidence of 0.136 ie 13.6%.

```
# the confidence of prediction of next city when the search is Chicago IL for fp-tree trained on all countries

city = {'Chicago IL'}
match = rule[rule['antecedents'] == city]
print("The next possible city to be searched:", match.loc[match['confidence'].idxmax()]['consequents'])
print("The confidence of predicscore:",match.loc[match['confidence'].idxmax()]['confidence'])
```

```
city: frozenset({'New York NY'})
score: 0.1364325416877211
```

The code is present in ***Assignment_association_rule_mining_onlyUS.py*** in which I am trying to grow a tree for just US.

Question 4: How did you measure the performance of the prediction algorithms from questions 2 and 3? What is your confidence that the measured score is accurate?

I analysed the confidence of prediction and also the execution speed. The measured scores can be verified with manually analysing the patterns and also by collecting the feedback from the customers using the website.

References:

Han, Jiawei, Jian Pei, Yiwen Yin, and Runying Mao. "Mining frequent patterns without candidate generation. "A frequent-pattern tree approach." Data mining and knowledge discovery 8, no. 1 (2004): 53-87.

Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." Proc. 20th int. conf. very large data bases, VLDB. Vol. 1215. 1994.

Solution to data challenge by Susha Pozhampallan Suresh for Machine Learning Engineer in Coveo

Raschka, Sebastian (2018) MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack.