HAMS RECRUITMENT CHALLENGE

Data Analysis Report

FRANCESCO MORAGLIO

July 6, 2020

1 STRUCTURE

An example of solution to your challenge has been implemented in four python scripts; the working of each is carefully explained through comments inside the code. In this paper I provide a general description of their behavior and of the results they produce (including the plots shown here). Suggestions on possible further analysis of the data are included throughout the paper.
Scripts can be found in the root folder and are named `data_transformation.py` , `general_statistics.py`, `user_clustering.py` and `user_journey.py`. Folder `data` contains raw data you provide for the challenge, while other `_data` folders are for storing results computed by the scripts.
Finally, file `analysis.pbix` contains some dynamical data visualizations made with Microsoft Power BI.

2 DATA TRANSFORMATION

For most of the statistical analysis that is performed in my solution, sanitization of the data was required. This transformation is also useful for increasing the scripts' performance, as querying the tables systematically by making use of `pandas` library is computationally expensive.
Such process is implemented in `data_transformation.py`, which outputs two tables: `user_table_full.csv` and `time_channels_table.csv`. Each row of the former table corresponds to a single user and contains the total revenue over time, fractioned by channel (taking into account the `IHC` Index). Second table, instead, contains the date-by-date total revenue over users, again fractioned by channel.
Note that, in both cases, the computation of value matrices is implemented to speed up execution, since the write functionality of `pandas` over DataFrames appeared to be too slow.

3 GENERAL STATISTICS

Few examples of summary statistics that can be computed are provided through the script `general_statistics.py`. Among these, one of the most relevant is that six specific channels, namely "A", "B", "E", "G", "H" and "I" can be considered primary since together they generate around 91% of the total revenue, as can be seen in figure 1.

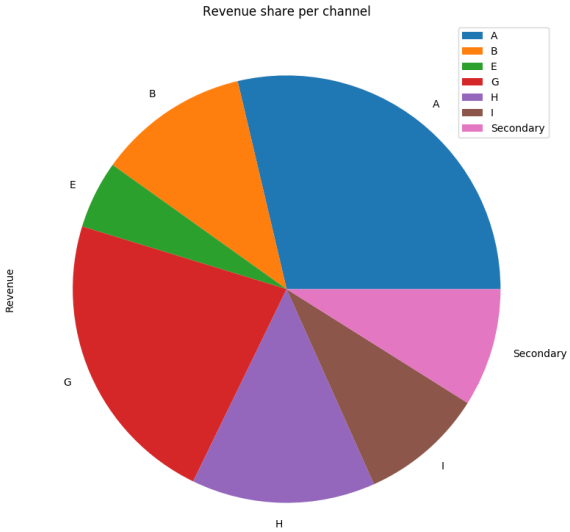


Figure 1: Channel influence over users

Computing the average spending per user in each of these primary channels, it can be found out that such values for channels "A" and "G" are significantly higher than elsewhere. Another insightful result comes from computing the correlation matrix of channels: they are essentially all independent; a slight correlation of around 21% is computed among channels "A" and "B".
`time_channels_table.csv` can be considered as the multivariate time series of revenue per channel, and some graphical representation of these data are given in the code. Peaks are clear and tend to be common to main channels; further analysis of high-spending periods could indeed be performed, together with employing this table for predictions using time-series specific algorithms. This wasn't included in my solution since I focused more on other aspects.

4 CLUSTER ANALYSIS

Most interesting results of this small analysis came out from clustering, basing on the relative spending per channel.
This has been implemented in script `user_clustering.py`. Considering the "large" dimensions of the dataset (relative to the low-spec hardware of my computer), I went for the K-means algorithm implementation of library `scikit-learn`, that is multi-thread optimized.
Appropriate number of cluster was determined to be $K = 2$, according to Silhouette method and Calinski-Harabasz score method. Davies-Bouldin index suggested instead $K = 3$, but this analysis was not performed since results appeared to be satisfying with two clusters.
Principal Component Analysis with two components was performed on the data to obtain a visual representation of the clusters, that is reported in figure 2. Note that, in this case PCA is good only for visual representation; attaining a good dimensionality reduction (e.g. to speed up machine learning algorithms) would require at least 15 principal components.
Having a look at summary statistics justifies considering users in cluster 0 as "average-

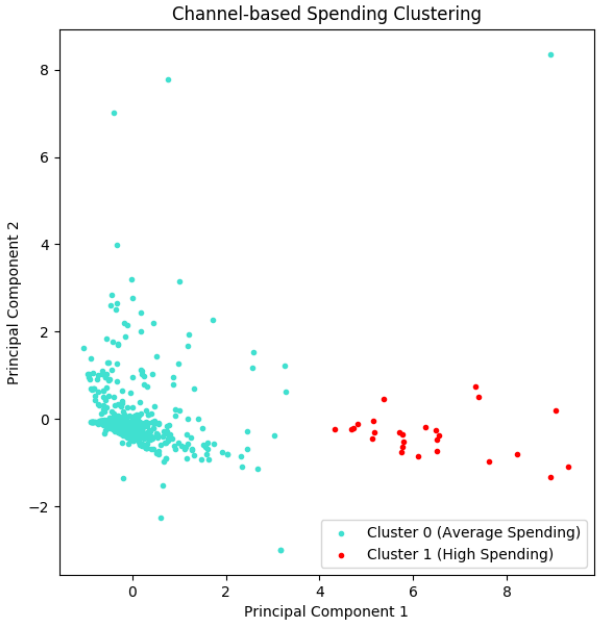
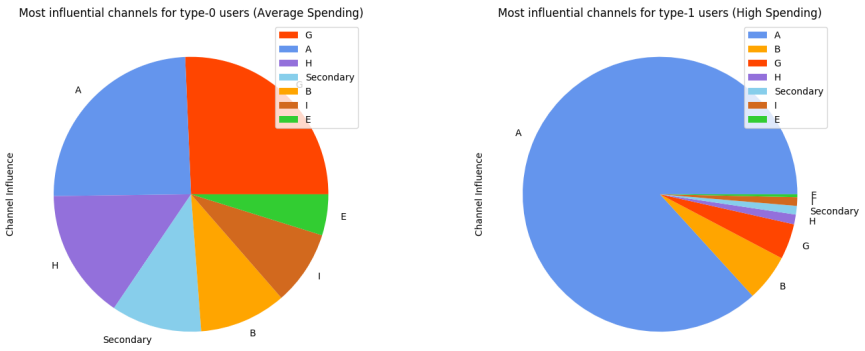


Figure 2: Cluster representation of a random sample of 1000 users.

spending", while those in cluster 1 can be regarded ad "high-spending". An interesting fact: type-1 users are around 2.7% of total users and produce more than 15% of total revenue (and I'd say it's worth considering them).

Analyzing channel influence over users in the two clusters also produces interesting insights. Values are represented graphically in figure 3. In particular, for the great majority type-1 users, channel "A" is the most significant; channels "G" and "B" also have some relevance, while others appear to be less important. This is probably related to results in Section 3 ("A" and "G" have higher average spending; "B" is correlated to "A"). This kind of analysis could be useful for targeted advertising, e.g. using the prediction features of clustering methods.



(a) Cluster 0. **(b)** Cluster 1.

Figure 3: Channel influence in each cluster.

5 CUSTOMER JOURNEY

Last script of the solution is named `customer_journey.py`. Inside it one can find the implementation of a class for investigating user-specific statistics, plus an example of usage. An instance of such class, named `Customer_Journey`, takes as input a User ID and generates some descriptive values, such as the customer's number of conversions, whether he/she is a return customer, his/her average spending and so on. Moreover, it features some methods that I explain briefly below.

- `compute_journey_attributions()` generates a dataframe with the detailed information about the attributions of the user, taking into account dates so that it is possible to reconstruct the customer's journey.
- `statistics_summary()` prints summary information.
- `channel_relevance_visualization()` generates and plots a pie chart describing channel relevance for the given user.
- `attribution_visualization()` generates and plots a chart that sums up the customer's journey. Each conversion over time is represented by a bar plot (representing the relevance of each channel for that conversion). An example of this method can be seen in figure 4.
- `write_journey_table()` writes the journey attribution dataframe in `csv` format.

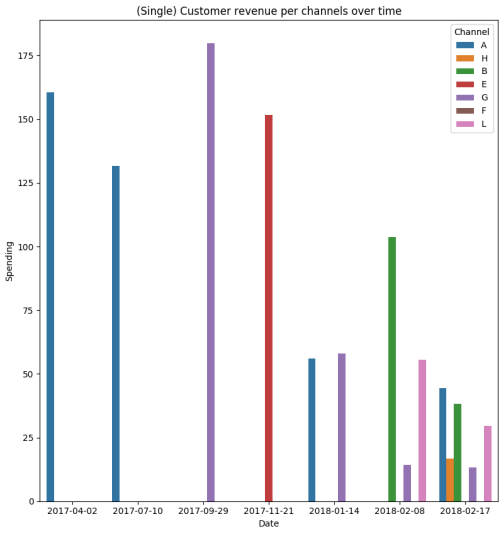


Figure 4: Sample plot of customer journey

Besides investigating a specific the journey of a specific customer, this class could be employed to generate useful aggregate statistics. For instance, it could be used to determine KPI's as the fraction of return customers or to build more advanced interaction phase models.