Customer Segmentation Project
Week 9

Data Cleansing and Transformation

Name: Customer Segmentation Project
Report date: 20-Sep-2021
Internship Batch: LISUM02
Specialization: Data Science
Group Name: Data Explorers
GitHub link: https://github.com/joeanton719/Customer-Segmentation-Project

Team member's details:

- **Joseph Antony**
    - Email: joeanton719@gmail.com
    - Country: Turkey
    - Company: UrbanStat
- **Melisa Gözet**
    - Email: mgozet@gmail.com
    - Country: Turkey
    - University: Ankara University
- **Dilem Ünal**
    - Email: diilemunal@gmail.com
    - Country: Turkey
    - University: Istanbul Aydın University
- **Aynur Cemre Aka**
    - Email: cemreaka@gmail.com
    - Country: Turkey
    - University: Yaşar University

Problem description

XYZ bank wants to roll out Christmas offers to their customers. But the bank does not want to roll out the same offers to all customers. Instead, they want to roll out personalized offers to specific set of customers. Manually trying to understand the customer categories would not be efficient and they will not be able to uncover the hidden pattern in the data (pattern which group certain customers in one category). The bank approached ABC Analytics company to solve their problem. The bank also informed ABC Analytics that they don't want more than 5 groups as this will be inefficient for their campaign.

ML Problem: ABC Analytics is to identify at least 5 customer segments from the provided data using Unsupervised Machine Learning for XYZ Bank.

ABC Analytics has assigned this task to their analytics team and instructed them to come up with an approach to identify and group customers into categories having similar attributes.

Initially, the dataset provided contained 1,000,000 rows and 47 columns. The next sections will provide a summary of data cleaning, data transformation and feature engineering done on the dataset.

1. **Renaming Columns**
   All the column names were translated from Spanish to English.

2. **Dropping redundant features**
   The first two columns (Unnamed: 0 & Date) were dropped as these columns did not provide any useful information. Also, we noticed two columns provided the exact same information (cod_prov & name_province). Both variables were the same since name_province is the name of the Spanish provinces and cod_prov is the number assigned to these provinces. Hence, cod_prov was dropped.

3. **Deduplication**
   Checking for duplicate values showed us that there were about 373,841observations with duplicate customer codes. Since the bank wants to segment unique customers, we dropped observations with duplicate customer codes and kept only the last observation for each customer code. This reduced the number of observations to 626,159. The customer code column was subsequently dropped as this column had only unique values and does not provide any more information.

4. **Handling Missing Values**
   Two columns (last_date_primary_cust & cust_spouse_index) were identified to have over 99% missing values. There were immediately dropped. This reduced the number of columns from 47 initially to 42.

   Furthermore, we identified 6,985 observations across most of the columns having exact same indexes with missing values. This indicates that some of the data are **Missing Not at Random (MNAR).** Since it would be impossible to impute these observations with meaningful information as other columns belonging to same observation has missing values, these missing values were dropped row-wise.

For the rest of the columns, missing values were assumed to be **Missing at Random (MAR),** and these can be imputed using information from other columns. A brief explanation is outlined here:

- **channel_to_join (categorical)**: All observations with missing values are from those customers from Spain. Therefore, this was imputed with the mode of those customers from Spain living in MADRID.

- **name_province (categorical):** There were 3,682 missing observations. Most of the customers were not from Spain. Therefore, this column was imputed with the mode of the category of customers living outside of Spain.

- **household_income (continuous):** There were 104,727 observations with missing values. We noticed that the household income for each of the Spanish provinces differed. As the household across all these provinces were heavily skewed, we imputed this column by filtering the observations by province and then taking the median income of that province.

- **Payroll & pensions2 (categorical):** Finally, both these columns were imputed with their own mode as both columns were binary.

5. Extreme Values

cust_seniority column has two observations with an extreme value of -999999. These observations were dropped.

6. Outlier Engineering

- **Age**: This column had customer ages as young as 2 years and as older as 116 years. Ages below 20 were grouped together and assigned a common age of 18. Similarly, Ages above 85 were grouped and assigned a common age of 85. These has led to a high frequency of 18 and 85 years at either ends of the Age distribution. When visualizing customer products by age, we noticed ages below 20 had the same proportions of products. Similarly, ages above 85 all has similar proportion of products.

- **household_income:** The household income was heavily skewed to the right. Hence, we applied two step outlier engineering techniques. First, we applied BoxCox transformation to transform the variable to nearly normal distribution. This led to the variables to have outliers at either ends of the distribution. Subsequently, we applied winsorization, which helped re-distribute outliers to either ends of the variable. As a result, there is a high frequency of observations at both ends of the variable.

7. **Feature Engineering:**
   New features have been created based on certain columns:

   - **first_holder_date:** As clustering algorithms cannot work with datetime object, we extracted year, month, and weekday from this column.

   - **total_accounts:** A new column called "total_accounts" was created by summing all bank products row-wise, thereby showing how many accounts a customer holds. There are customer's who hold accounts as low as just 1 single account and as high as 14 accounts.

   - **dup_rows_count:** Even after dropping around 300K supplicated observations at the beginning, we still found the dataset has around 12,915 duplicated rows. However, this indicates that there are around that many customers having same attributes. Therefore, we created a new column called "dup_rows_count" that has the number of duplicates for rows having the same attributes. Subsequently, all duplicated rows were dropped. This column can potentially be used as weights for the kmeans algorithm.

8. **Handling Cardinality / Categorical Encoding**
   Many columns have high cardinality, with over 100 categories. Additionally, as most clustering algorithms requires all inputs to be numerical, we had to use suitable encoding techniques to convert categorical variables to numerical. We have outlines what we have done below:

   - **cust_residence**: This variable had over 100 countries. We grouped this variable by continent, thereby reducing the number of categories from 113 to just 7 categories (ES was kept separate and not grouped with EU continent).

   - We created a pipeline that carries all necessary encoders and transformers (including BoxCox transformation, and Winsoriser for household_income variable). For those variables with fewer categories (emp_index, cust_gender, cust_rel_time, cust_residence), One Hot Encoding will be applied which will transform each category within the columns to its own columns with binary inputs. Columns with high cardinality (name_province, channel_to_join) will be encoded with CountFrequency encoder, which will turn each category to its frequency within that column.

With this, we have cleaned the whole dataset and created a column transformer pipeline that will further help with modeling. The **final preprocessed dataset contains 606,227 observations and 45 columns**. **Applying the column transformer** to the preprocessed dataset **will increase the number of columns to 56**.