DATANS

Grouping of sections for a physical cell phone store.

Technical Documentation

Daniel Fragoso Alvarado, Jesús Enrique Gómez Martínez & Hugo Rangel Ramírez

IIMAS, *UNAM* — May 25th, 2022

1 Problem

Currently, there is an enormous variety of cell types, either because they vary in size, brand, operating system, or some other characteristic. This generates a chain of problems for department stores, for example, the poor distribution of their products on the displays and inventory problems such as the lack or overload of certain equipment, which often ends in the loss of money due to delayed inventory, or of customers for not meeting their needs.

Furthermore, a bad grouping of cell phones hinders the purchase of the same or of equipment that is similar to those initially sought, which is why a correct clustering of cell phones based on the similarity of their characteristics would increase store sales and decrease sales. their losses. But how can we carry out a correct and efficient grouping of cell phones?

Data The dataset used for the model generation, was extracted from the Kaggle discussion. The dataset set contains data about the mobile phones which were released from 2012 to 2021, and which can be bought in Ukraine. In total the dataset has 1224 records and 13 columns; that contains the model name, brand name and operating system of the phone and it's popularity. It also has it's financial characteristics like lowest/highest/best price and sellers amount. And some of the characteristics like screen/battery size, memory amount and release date.

To work with this dataset it is necessary to correctly identify what properties and problems it may have. The main feature is the fact that it presents categorical variables (such as brand and operating system) as well as numerical variables (popularity, screen size) and variables that can be treated both numerically and categorically (memory capacity), so a classic clustering approach isn't enough to solve the problem. In addition, there are inconsistencies with the units in which some parameters are recorded, as well as the lack of some attributes for some records (null) and scattered data that complicate the clustering task.

2 Problem Modelation

We came up against the problem of classification of mixed data; categorical and numerical features. One of the most popular algorithms that is used for solving that problem is *k-prototypes*, which is a combination of the *k-means* and *k-modes* algorithms, the first one it is useful for numerical classification, and the second one for categorical features.

The k-prototypes algorithm uses a dissimilarity function between two mixed-type objects. Let $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, which their first p variables are numerical and the following $m - p$ variables are categorical, then we define $d(X, Y)$ as follows:

$$d(X,Y) = \sum_{j=1}^{p}(x_j - y_j)^2 + \gamma \sum_{j=p+1}^{m} \delta(x_j, y_j) \qquad \text{where} \quad \delta(x_j, y_j) = \begin{cases} 0 & (x_j = y_j) \\ 1 & (x_j \neq y_j) \end{cases}$$

and γ is a weighing factor that determines relative importance of numerical versus categorical features.

Now, let k the number of clusters, $X = [X_1, \dots X_n]$ a set of n objects, $X_i = (x_{i,1}, \dots x_{i,p}, x_{i,p+1}, \dots x_{i,m})$; $x_{i,1}, \dots x_{i,p}$ are numerical and $x_{i,p+1}, \dots x_{i,m}$ are categorical. So the cost function that the k-prototypes algorithm try to minimize is:

$$P(W,Q) = \sum_{l=1}^{k} \left(\sum_{i=1}^{n} w_{i,l} \sum_{j=1}^{p} (x_{i,j} - q_{l,j})^2 + \gamma \sum_{i=1}^{n} w_{i,l} \sum_{j=p+1}^{m} \delta(x_{i,j}, q_{l,j}) \right)$$

What the algorithm does is initialize Q^0, and then iteratively solves the problems P_1 and P_2 until no object has changed clusters after a full cycle test of the whole data set, or the pre-defined number of iterations is exceeded.

- Problem P_1: Fix $Q = \hat{Q}$ and solve the reduced problem $P(W, \hat{Q})$
- Problem P_2: Fix $W = \hat{W}$ and solve the reduced problem $P(\hat{W}, Q)$

The problem P_1 can be solved as follows:

$$\begin{aligned} w_{i,l} &= 1 \quad \text{if} \quad d(X_i, Q_l) \leq d(X_i, Q_t), \quad \text{for} \quad 1 \leq t \leq k \\ w_{i,t} &= 0 \quad \text{for} \quad t \neq l \end{aligned}$$

And for problem P_2, the solution of $q_{l,j}$ for numerical features is the mean relative to the l cluster of the j feature, and for categorical features is the mode relative to the l cluster of the j feature.

3 Application

The implementation of the k-prototypes model can be found in Python, through the k-modes library; and which can be imported by means of the pip command to our work environment: `!pip install k-modes`. As already mentioned, the k-prototypes algorithm combines k-modes and k-means and is capable of clustering mixed numerical/categorical data, and internally uses Numpy as one of its main processing and classification tools.

Pre-processing Before using the model, we must do some pre-processing with the data. First the selection of relevant features. From the 12 categories of the original dataset, only 6 were selected: The brand name (`brand_name`), operating system (`os`), the recommended price (`best_price`), the screen size (`screen_size`), the memory size (`memory_size`), and the battery size (`battery_size`). 3 categories and 3 numerical.

As for the missing values, for the categorical categories such as brand, operating system, and memory, we decided to replace the null values with the mode, while the numerical values were replaced with the median. In addition, in the case of the memory category, there were incongruent values such as a memory size of 0.0032. Therefore, these values had to be replaced by their equivalent but according to the section. For example, from 0.0032 to 32. These conversions were made taking into account the unique values of the rest of the category.

On the other hand, the data corresponding to battery capacity (`battery_size`), screen size (`screen_size`) and price (`price`) are numerical values with a considerable range, which can cause problems in the accuracy of the model. Therefore, we decided to normalize these categories using the min-max method.

The `MinMaxScaler` function from the sklearn.preprocessing library was used. Once imported, 3 normalization functions were trained, one for each data category, because each category has a different range of values to be normalized. Once trained, the data for each category were transformed. The normalization functions will be used to transform the data entered by the user in order to generate the prediction.

Model Application With the previous processing, we have a database that is functional for the use of the algorithm and the generation of the groupings. Therefore, we proceeded to define a class of type `KPrototypes`, which receives a particular parameter: the number of clusters. In addition, to make it reproducible externally to our workspace, a seed was defined to always generate the same selections of random clusters, by `random_state = 2022`.

Once the class is started, the next step is to train and predict the clustering of our data, for which we need to make use of the `.fit_predict` function. This function receives the data, as well as a list indicating the index of the columns that are categorical. The number of clusters was set to 25. It should be clarified that under the elbow method that is usually used in the k-means algorithm, in order to determine the number of clusters, the result (only entering the numerical characteristics) was 3. For the application we are trying to implement, a small number of clusters leads to the very problem we are trying to solve, an excess of choices for buyers.

Therefore, it was decided to generate several predictions by increasing in each one the number of clusters required, and to observe the number of cells grouped by each cluster in each one. At the end of generating several tests, the most homogeneous classification, in terms of the number of cells in each cluster, was the one for which 25 was defined as the number of clusters.

Generation The model's `classification` function first verifies that there is a record of the categorical variables, since our model is based on a historical record, so new features must be added manually, once this filter has been passed, it normalizes the numerical variables with the training before described and the main model is applied to that new record, returning the cluster to which it was assigned and automatically updating the database to include this new equipment along with its cluster.

On the other hand, the `recommendation` function takes the properties given by the client, if some feature is provided for which there is no record or simply omits it, the system uses by default a registered feature according to the type of this property, for example , for categorical variables it will use the mode of the records of that attribute, while for numerical variables the median is chosen because it is the one that has less sensitivity to outliers, once this is done the variables are normalized and the main algorithm is applied for determine the cluster in which the customer is most likely to satisfy their purchasing needs. Finally, through the `filter` attribute, you can indicate to which you prefer to give greater weight to the characteristics so that the function automatically returns the recommendations ordered by that attribute.

Results Finally we are going to generate simulate some queries to our functions. The complete result is too extensive, so it can be consulted in the code file inside the .zip that comes attached to the delivery of this document.

```
Recomendacion(nombre_marca='Honor', os='Android', memoria=128, ascending=True,
    filtro = 'battery')
```

```
 >
+----+--------------+-----------------------+---------+--------------+---------------+-----------+
|    | brand_name   | model_name            |   ...   |   memory_size |   battery_size |   cluster |
|----+--------------+-----------------------+---------+--------------+---------------+-----------|
|  0 | Sony         | Xperia 10 I4113 Black |   ...   |           64 |          2870 |        16 |
|  1 | Sharp        | Aquos D10 4/64GB Black |   ...   |           64 |          2900 |        16 |
|  2 | Honor        | 9i 4/64GB Blue        |   ...   |           64 |          3000 |        16 |
+----+--------------+-----------------------+---------+--------------+---------------+-----------+
```

```
Recomendacion(nombre_marca='Huawei', os='Android', memoria=64, ascending=True,
    filtro = 'memory')
```

```
 >
+----+--------------+-------------------------------------------------+---------+--------------+-----------+
|    | brand_name   | model_name                                      | os      |   memory_size |   cluster |
|----+--------------+-------------------------------------------------+---------+--------------+-----------|
|  0 | Samsung      | Galaxy A12 SM-A125F 4/64GB Black (SM-A125FZKVSEK) | Android |           64 |         0 |
|  1 | AGM          | X3 6/64GB Black                                 | Android |           64 |         0 |
|  2 | CAT          | S61 Black                                       | Android |           64 |         0 |
+----+--------------+-------------------------------------------------+---------+--------------+-----------+
```

```
Clasificador('OnePlus', 'OxygenOS', 7000, 6.1, 128, 2700)[1]
```

```
 > [1]
```

4 Conclusions and future Work

In conclusion, our model meets the main objective, which is the classification of cellular equipment by its components, it provides a considerable number of clusters with a uniform distribution that allows the correct physical grouping of them to optimize the sales and recommendation process.

The next thing to do is the creation of a graphical interface with a user-friendly design that facilitates the use of the algorithm and its extension to an application that can be integrated with the digital resources of department stores. In addition, the algorithm is reproducible in other aspects both within the world of technology and outside of it, for example using the same algorithm applied in laptops, speakers, etc. Based on the same principles of hardware and software features. Or in diet plans and recipes that are based on the ingredients or nutritional contributions.

On the other hand, it is also possible to optimize the manual registration of new attributes or extend the capacity of the algorithm so that it is capable of making comparisons between given products.

References

de Vos, N. (2022, 05). *GitHub - nicodv/kmodes: Python implementations of the k-modes and k-prototypes clustering algorithms, for clustering categorical data.* Retrieved from https://github.com/nicodv/kmodes#huang97

Huang, Z. (1998). Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery*, *2*(3), 283–304. doi: 10.1023/a:1009769707641

Ruberts, A. (2020, 05). *K-Prototypes - Customer Clustering with Mixed Data Types.* Retrieved from https://antonsruberts.github.io/kproto-audience/