Monjoux Hugo
01/09/2020

Report for the technical test

Exercise 1A

The context for this exercise regards the usage of the ExpiryApp that helps teams in stores to manage expiration dates of products. When a product is added to the App, the expiration date needs to be set and the app will then keep track of all the products. The problem that is addressed here is that the products need to be manually added to the ExpiryApp product list. Sometimes products can be forgotten.

A script is therefore needed to help find thems in the store product list to recommend them for the ExpiryApp.

To answer this problem it is necessary to first understand the data at hand and how to exploit it. Then we will discuss the approach used. In the end we will take a look at the improvements that can be made on the script.

1. Data Analysis

For this exercise we have access to two datasets:

- A fake set of products currently tracked by our system in a specific shop (3114 products)
- A fake extraction of the shop internal system which contains its vision of the existing references and the current shop assortment (39136 products)

Each product tracked in the ExpiryApp system as the following fields associated with him: shop name, aisle, bar code reference (EAN), the name of the product and the last time it was checked. The Shop internal system tracks many more fields for each products. For example it tracks the Label (a common name used to refer to a category of products) and the quantity still in stock.
The two datasets can be joined on the column EAN present in the two. By joining them we can start our analysis.

First, it is interesting to see how many Labels are defined:

Label	**Count**
Famille	66
Sous-Famille	313
Unité de Besoin	1489
Code Interne Enseigne	33524

As you can see the more the Label is precise, the more the count of different Labels is high. This means that we can group every product defined in a specific category and we can find a product based on the hierarchy of labels: Famille > Sous-Famille > Unité de Besoin > Code Interne Enseigne > Product Name.
Now, if we compare this result with the repartition of labels in the products tracked we get the following table:

Label	**Count in Shop dataset**	**Count in App dataset**
Famille	66	40
Sous-Famille	313	177
Unité de Besoin	1489	504
Code Interne Enseigne	33524	2931

Some labels are not represented in the App, we can imagine that the user decided to track them by other means. It is also interesting to take a look at the spread of products by Label in the App. In fact it may be possible that some types of products appear only once or twice. The following table presents a summary of this study:

	Famille	**Sous-Famiille**	**Unité de Besoin**	**Code Interne Enseigne**
Mean	74	16	5	1
Standard Deviation	73	21	7	0
25%	12	3	1	1
Min	1	1	1	1

For each label we counted the number of products tracked and computed the mean, standard deviation, minimum value and first quartile. As you can see some families have only one product defined on the App. What's more the standard deviations are really big, which means that there is a big difference between the number of products of the different classes. It also shows that Unité de besoin and Code Interne Enseigne are too specific to give us any information on the product repartition.

Another interesting field defined for each product is the 'Quantity in Stock' field. It is a numerical value that tracks the number of products in the shop. The value can be positive or negative. When negative it indicates that the product was a pack and only an item of the pack was taken. When the value is null it means that even if the product is defined in the system there is no such product in the shop. A quick statistical analysis showed that the field as a wide range of values, positive and negative but the mean quantity is of 4.

The Etat assortiment fields keep track of the displayed products in the shop. If a product is not displayed it is defined as Hors assortiment. Such a product may expire but is not at risk to be sold to a shopper.

The last fields we will take a look at are the last reception field in the shop system and the last_control in the App data. The last_ control field corresponds to the last time the user updated the values of the product in the app (he checked if the product was still good). In our dataset the timerange goes from 01/2020 to 08/2020. On the other and the date of the last reception corresponds to the last time a product was received by the shop. It can go from 06/2015 to 08/2020. In our problem there may be no need to consider products received too long ago as they may have been left out on purpose.

Now that we have a better understanding of our data we will take a look at how we can answer our problem.

2. Approach

The environment in which the script runs is a python 3.7 environment with the pandas and xlrd libraries installed. As explained in the first part we have two data sources, both contain pertinent information. Pandas DataFrames were used as the base structure to work with. The csv and xlrd, once imported, are therefore stored on the same format.

The problem to be tackled by this script is : Is it possible to find and recommend the most pertinent products that would need to be inserted in the ExpiryApp. I divided this problem in two parts:

- Is it possible to find all the products that should be on the Expiry App?
- Is it possible to select only the most probable ones?

To answer the first question it was necessary to filter the products listed in the shop system without missing any. The filters used are based on the result of the data analysis done before, they are (in order of usage in the script):

- Filtering the product by dividing the products already initialized on the app and those that are not. From the initial 39136 only 36139 remain.
- Filtering the products by keeping only the Sous-Famille Labels seen in the ExpiryApp. They are general enough so as to not impede new products to be added to the app. Only 23911 products remain.
- Filtering the products by only keeping the products that are in the shop (the quantity in stock needs to be different from 0). Only 2023 products remain.

The number of products that we obtain is too big considering the work that needs to be done by the user. It is also interesting to note that of all these products only 30 were received before 2020 and none before 2019. We will conjecture that these products are still used by the store as they are not older than one year.

We now need to answer the second question and to find the most pertinent products to initialize. The steps used to do this are the following:

- Filtering the products based on their Etat d’assortiment. We hypothesize that the products not displayed will not be a high priority for the app user when checking expiration dates. We can therefore remove them from our list. From the 2023 products 1806 remain.
- The remaining products are sorted by the labels Famille and Sous-Famille which have the most representation in the Expiry app.

We now have a relatively short list of products that our user would most probably initialize in the App.

For the optional objective the aisle name was taken from the field aisle defined in the ExpiryApp and associated with the Sous-Famille label so every product has an associated aisle.

3. Conclusion and Improvements

The resulting script answers the problem that we had while making some assumptions on the preferences of the user. Some improvements could come with letting the user choose which filter he wants to use.
What's more, some errors in the datasets can be present and not addressed with this script. For example the quantity in stock could be negative while the product is not a pack. Another type of error is present when doing the join between the in app data and the shop dataset. Some products defined in the app are not present in the shop. This type of error is not yet handled by the script.

Exercise 1B

In this exercise we consider the following context:

When a product is near it's expiration date, ExpiryApp will notify the user. There are different ways to handle the soon expired product. The user could put a discount sticker on it, or maybe give it to a charity or a association that could use some products.
Would it be possible to make the ExpiryApp suggest a solution to optimize this choice?

To answer this problem I would proceed in three steps.

1. Data Collection and definition of needs by the clients

The first step in every data science project is to work on the data available for the project. As such the first thing to do would be to study all the data available to us and gather more if necessary. For this kind of project some data needed would be the price of the product, the seasonality of the product, where the shop is situated, what time of the year it is, what type of clientele frequents the shop usually, etc…
What's more, there may be a need to restructure the data as the raw data may not be exploited easely.
To collect, structure and study the relationship between those is the cornerstone of this step. To do this efficiently, the needs of the clients need to be assessed as best as possible. Thus enabling the data scientist to have a better idea of the context in which the project exists.

The tools used for this step are: a development environment running with python to leverage the multiple libraries of data visualization, data mining and exploratory data analysis such as pandas, numpy, matplotlib, seaborn….
The database also needs to be able to contain the data to be used. Depending on how the data is structured, a NoSql or Sql might need to be put in place (examples of those are Elasticsearch, MongoDb, PostgreSql, MySql).

2. Prototyping a predictive model

The second step in these project would be to model the problem. This implies that a cost function is necessary. It should make it easy to understand if the model gives the best answer or not. In the context of these problem, a metric that could show the performance of the model is how much money was made with a solution while another could be how many products were wasted with these solution. A huge part of this step lays on how the cost function is defined as it will serve as the reference between all the models tested and make sure the right one is chosen.

Depending on how the data is gathered there are multiple solutions in regard to the architecture of the predictive model.

The first one would be divided in two different models. A prediction of the sales for the products and a choice between the outcomes when the expiration date is considered. The prediction of the products sales is a prediction of a continuous value and as such should be treated with algorithms like Random Forests, SVM (support vector machines), linear regressions and if interpreted as a time series problem RNN (recurrent neural network) could be used. Then the choice of action is a classification problem. The algorithms to test are SVMs, Random Forests, logistic regressions and CNN (convolutional neural networks)

The second solution would be to consider that the problem is a classification problem from the beginning and leverage the same algorithms to create a model.

In both cases the training dataset and test dataset should be big enough to give plausible results.
Once a fitting model is found, it can be improved by checking it's parameters. A grid search will make it possible to test each parameters against each other to find the best possible combination for the model.

The tools used for this step are: a python environment with the scikit-learn, tensorflow, keras, pandas libraries

3. Putting the model in operations

Once it is possible to answer our problem with a model, it needs to be put into operations. To choose the best way to implement it depends on the size of it. If it is too computationally heavy it may not run on the user's application. In that case it may be better to run it on a server or to run the model with common products so as to create a database of results that can be easily accessed by the user. This step depends heavily on the two previous steps.

This project is complex and depending on how the data is gathered could take more time. For this type of project, 5 to 6 months of development are necessary..