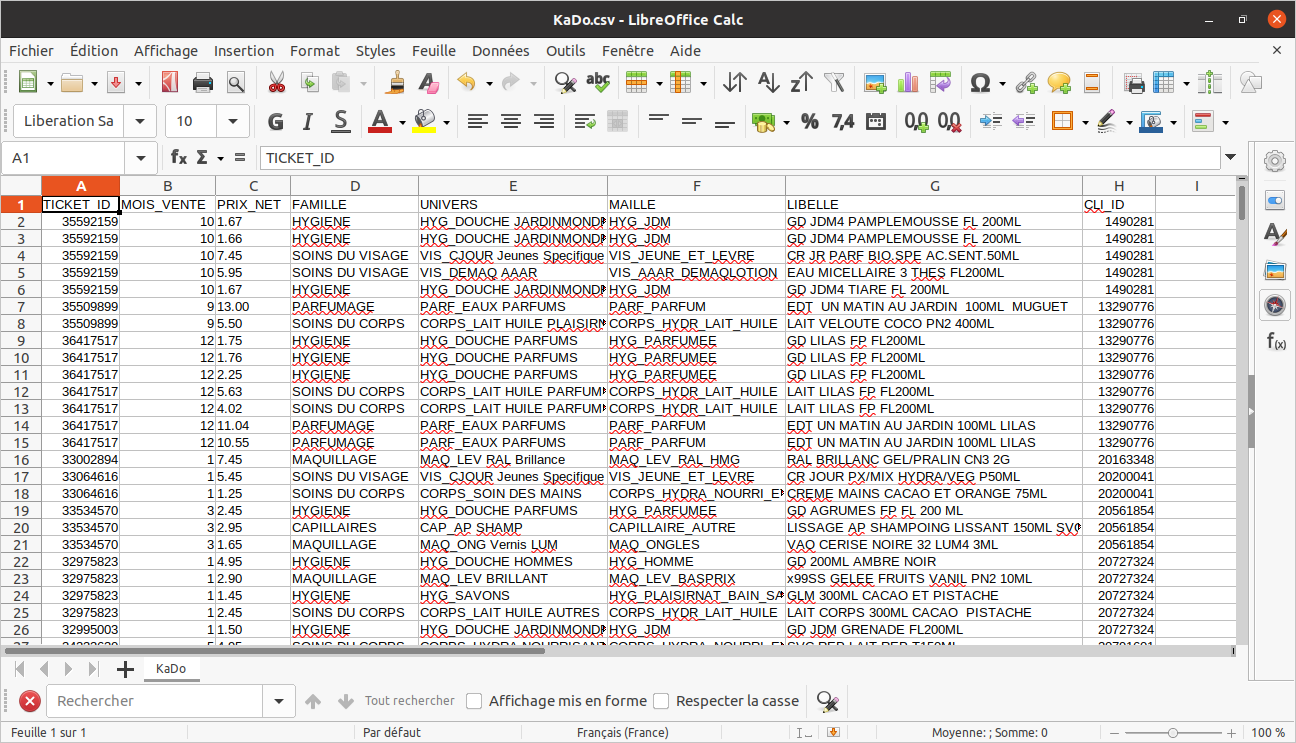


A company granted you access to KaDo: a database containing millions of bought items, divided in 3 categories. For instance, a bottle of red wine belongs to Famille: alcohol, Univers: wine, Maille: red wine.

**Dataset :**

The dataset contains more than 700,000 rows, illustrating all customer transactions in the company.

In order to provide a quick solution to show statistics on the global information, the development team proposes the PowerBi.

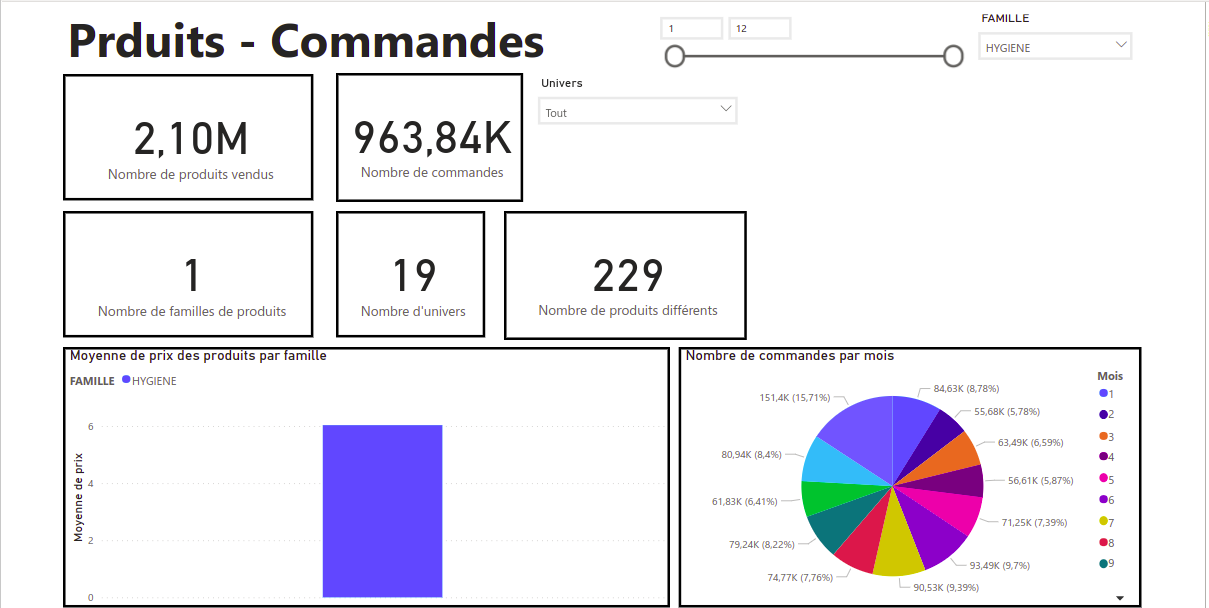
PowerBi, allows us from a CSV file, to have several dynamic and interactive graphs in order to have all the information necessary for the management of data with important quantity

**PowerBi :**

Power BI is a set of software services, applications and connectors that work together to transform disparate data sources into immersive and interactive visual information.







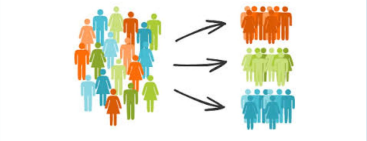
**Recommender System :**

There are many ways to build recommender systems for ratings-based data, such as movies and songs. The problem with rating-based models is that they cannot be easily normalized for data with unscaled target values, such as purchase or frequency data. For example, scores typically range from 0 to 5 or 0 to 10 for songs and movies.



The goal is to create a system to recommend products to customers using purchase data with Python and the Turicreate machine learning module. These steps include:

- Data transformation and normalization  
- Training of the models  
- Evaluation of model performance  
- Selection of the optimal model



When a customer taps for the first time on the "order" page, we can recommend the first 10 items to add to their cart, for example, hygiene products, vintages, etc.

The tool will also be able to search a list of recommendations based on a given user, for example:

- Entry: customer ID  
- Return: ranked list of items (product IDs) that the user is most likely to want to put in their (empty) "shopping cart".

I. Import modules

- pandas for data manipulation  
- turicreate for model selection and evaluation  
- sklearn for splitting data into training and test sets.

II. Load data

III. Prepare the data

Our goal here is to break down each item list in the product column into rows and count the number of products purchased by a user.

III. 1 Create data with user, item and target fields

- This table will serve as input for our later modeling.  
- In this case, our user is customerId, productId, and purchase\_count.

III.2 Create dummy data

- Dummy to mark whether a customer has purchased this item or not.  
- If someone buys an item, then purchase\_dummy is marked as 1.  
- Normalizing the number of purchases, for example for each user, would not work because customers may have different purchase frequencies and different tastes. However, we can normalize the items by purchase frequency for all users

III.3. Normalize item values across users

To do this, we normalize the purchase frequency of each item across users by first creating a user-item matrix as follows

In this step, we normalized their purchase history from 0 to 1 (1 being the highest number of purchases for an item and 0 being the zero number of purchases for that item).

IV. Separation of Training and Test Sets

- Splitting the data into training and test sets is an important part   
- We use an 80:20 ratio for our training and test set size.  
- The training part will be used to develop a predictive model, while the other part will be used to evaluate the performance of the model.

Now that we have three data sets with purchase accounts, dummy purchases, and scaled purchase accounts, we would like to split them for modeling.

V. Define the models using the Turicreate library

Before running a more complicated approach like collaborative filtering, we need to run a base model to compare and evaluate the models. We will use the popularity model.

A common approach to predicting purchased items is collaborative filtering. Let's first define our variables to use in the models:

Turicreate made it super easy for us to call a modeling technique, so let's define our function for all models as follows:

VI. Popularity model as baseline

- The popularity model takes the most popular items for recommendation. These items are products with the highest number of sales among customers.  
- The training data is used for model selection

VII.1. Similarity in cosine

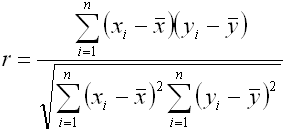
The similarity is the cosine of the angle between the 2 vectors of the item vectors of A and B  
The closer the vectors are, the smaller the angle and the larger the cosine.

It allows to quantify the similarity between 2 entities

**VII.2. Pearson similarity**

- The similarity is the Pearson coefficient between the two vectors.

The Pearson correlation coefficient, is an alternative method to normalize the count of common neighbors. This method compares the number of common neighbors to the expected value in a network where the vertices are randomly connected. This value is strictly between -1 and 1.



**VIII. Model evaluation**

To evaluate recommendation engines, we can use the concept of RMSE and precision-recall. Several evaluation criteria:

RMSE (Root Mean Squared Errors)

- Measures the error of predicted values  
- The lower the RMSE value, the better the recommendations.

Recall

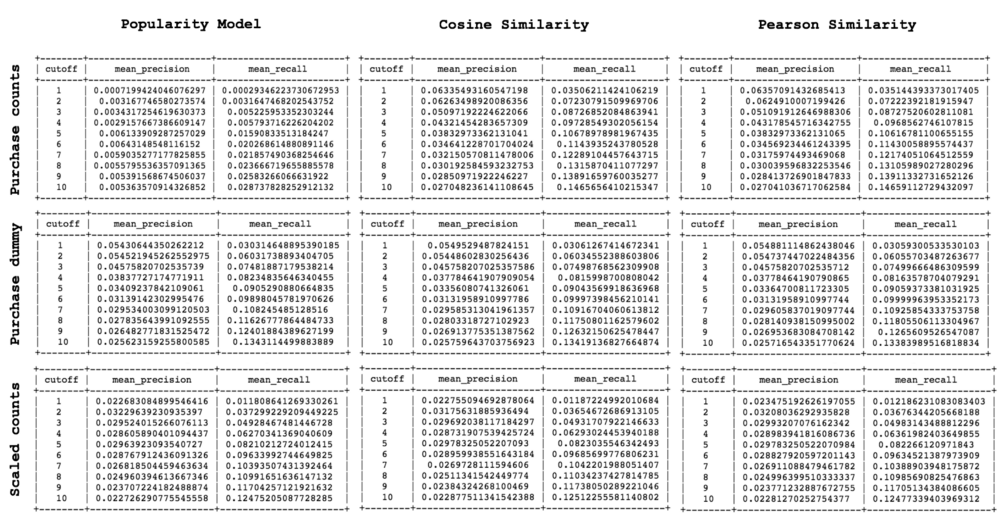
- What is the percentage of products a user buys that are actually recommended?  
- If a customer buys 5 products and the recommendation decides to show 3, the recall is 0.6.

Accuracy

- If the customer was recommended 5 products and bought 4, the precision is 0.8.

Why are recall and precision both important?

- Let's take the case where we recommend all products, so our customers will surely cover the items they liked and purchased. In this case, we have a 100% recall rate!  
- We need to consider accuracy. If we recommend 300 items but the user only likes and buys 3 of them, the precision is 0.1%! This very low precision indicates that the model is not excellent, despite its excellent recall.  
- So our goal should be to optimize both recall and precision (to be as close to 1 as possible).



**IX. Final output**

A csv file and then send it to the web application POWERBI to visualize its data.  
We can also have the results for only one custommer with his ID.

