Sweet, sweet Data Science

*Predicting factory demand of a bakery*

**By**

Miguel Ruiz Nogues

Illán Lois Bermejo

[Introduction 2](#_Toc29503599)

[Raw data 2](#_Toc29503600)

[Tech-suite 3](#_Toc29503601)

[Methodology & re-execution guidelines 3](#_Toc29503602)

[Re-execution steps 3](#_Toc29503603)

[Main issues encountered 4](#_Toc29503604)

[Project Results 5](#_Toc29503605)

[Conclusions & Improvements 6](#_Toc29503606)

[PROJECT CONCLUSIONS 6](#_Toc29503607)

[IMPROVEMENTS 6](#_Toc29503608)

[PERSONAL CONCLUSIONS 6](#_Toc29503609)

# Introduction

The following project consists in forecasting the orders of a product that the stores of a bakery chain order to the bakery factory.

The idea for this project came from the necessity of this bakery factory to obtain better forecasts of the incoming orders in order to better schedule the workforce and improve raw-materials procurement processes.

Currently, the methods used at the factory are based on expert knowledge and rules of thumbs. They store the historic data of their orders, however due to the lack of capabilities they have never analysed the data.

For this reason, the main goal for our project, is to showcase to the client (director of the bakery factory) the power of the data they currently own to leverage their business, by creating a predictive model for one of their products, and creating a dashboard to give them visibility of the past, present and future of the incoming orders.

We will be working on a snapshot of the data.

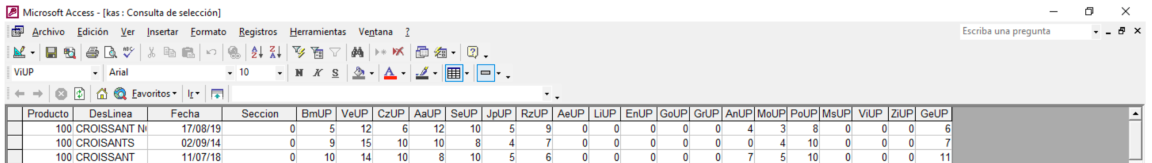
Tableau ADD PRODUCTS

Pasar documentos a PDF

# Raw data

1. *0\_original\_b2.csv*

The main dataset use was extracted from the Microsoft SQL Server of the client, containing orders from 01/01/2008 to 30/09/2019. The file contained 1607908 rows and 23 columns corresponding to the product id, description, date, section and store ordering that product (as displayed below):



It is important to note that the client provided us a second dataset named *0\_original\_b2.csv* containing orders from earlier, however it was not used due to age of the data.

1. *stores\_locations.csv*

Containing geographical information of the stores. The dataset was manually created based on the information of their website, and google maps coordenates.

1. *products.txt*

Containing the catalogue of offered products. The dataset was manually created from pdf catalogues hosted on their website, and it was used to cross-check product descriptions

1. Other Data Sources:

Other data sources where collected in order to study correlations with the time series, and add them as features, however due to the lack of time, and high autocorrelation of the time series were not utilized in the end:

* 1. Local festivities calendar
  2. Local football matches calendar
  3. Local weather

# Tech-suite

* **Main tech**: Python, Tableau & Github.
* **Non-standard-tech:\***: Scikit-learn, Git Large Files, statmodels, fbprophet, etc.
* **Standard tech \***: Pandas, Numpy, Seaborn, Matplotlib, etc.
* **Supporting tech**: Jira, Slack, Webex, Scikit-learn, virtual enviroments, pair programming, etc.

*\* Full list of libraries included in the repository: requirements.txt*

# Methodology & re-execution guidelines

In a nutshell, the project consisted in cleaning the very messy data provided by the client for 10 products, creating predictive models for one of the products, and a dashboard including data from the 10 cleaned products.

However, this is an ultra-simplistic summary of the project, let’s have a better look at the stages that we followed:

* **Data acquisition:** Data was provided by the client who stored it in Microsoft SQL Server, and before any cleaning was performed, the data was transformed to a transactions shape: date, product id, description, store & unit column.
* **Data cleaning & preparation:** Preparing the data was very painful, product descriptions didn’t necessarily correspond to product ids, and were manually written.
* **Analysis:** Before implementing predictive model, we conducted the following tests to gain better understanding of the behaviour of the time series:
  + Dickey-Fuller test for stationarity.
  + Autocorrelation function (ACF) and partially autocorrelation function (PACF) for lag weight analysis and ARIMA models components.
  + Series decomposition to analyse trend and seasonal components.
  + Liung-Box test for checking absence of autocorrelation in the residuals
  + Q-Q plot and Shapiro-Wilk test for checking normality in the residuals
* **Modelling:**

The models utilized where the following:

* + **Exponential Smoothing:** Selected because holds well for series with trends and seasonality.
  + **Arima:** Good model when the weight is found on the first lags. The MA() component makes it especially interesting for series with great residual values.
  + **Sarima:** Developed version of ARIMA with a seasonal component that can handle seasonality independently.
  + **Prophet:** Time series method developed by Facebook. Intuitive and powerful.
  + **Univariant Random Forest:** Machine learning method, tried out for comparison with traditional time series models.
* **Front-end & visualization:** To conclude the project, we created a visualization with the cleaned products, and a predictive model.

## Re-execution steps

To re execute follow the below-mentioned order:

1. Install all dependencies stated in the *requirements.txt* document.
2. Go to data -> notebooks and execute them in the following order:
3. Execute 00\_raw\_data\_to\_transaction\_data.ipynb, which transform data to a transaction shape.
4. Execute 01a\_all\_data\_introductory\_EDA.ipynb and 01b\_exploratory\_target\_products\_filtering.ipynb, which were created to gain more understanding of the data.
5. Execute 02\_data\_normalization\_and\_filtering.ipynb, which were created to filter the 10 products selected by the client.
6. Execute 03\_data\_cleaning\_and\_quality\_assurance.ipynb, which were created to filter clean the time series, and ensure the data made business sense.
7. Execute 04\_feature\_engineering.ipynb, which was created to include other features in the dataset, such as weather, football matches in Madrid,etc.
8. Execute 05-predicting\_palmera\_chocolate.ipynb, which was created to analyse the time series, and find the best predictive model.
9. Then, read the front\_end\_guide.pdf with instructions on how to execute the dashboard (note that it requires to initialize a tabpy server)

# Main issues encountered

* **Data cleaning**

Product Descriptions and IDs were a real mess. We found over 45.000 unique product descriptions (‘tarta’, ‘trta’, ‘tartita’, etc.) for 1500 unique products id, and the relationship between product id and product description was not tribal. In other words, under a product id X, there were orders for more than one product.

To be able to clean the data we had to:

* + Create a dataset of product descriptions based on their online-catalogue and using Levenshtein distance normalize all the words of product descriptions (‘tarta’, ‘trta’, ‘tartita’ = ‘tarta’)
  + Once the words were normalized, we tried - using fuzzywuzzy – to normalize the product descriptions in the dataset with the product descriptions in the catalogue, but results were not good.
  + We tried other methods, however, in the end, we cleaned the data for only the 10 relevant products, one by one, applying filters based on key-words on the dataset with normalized words.

* **Data reality**

Once the data was cleaned, we observed animalities that we could not explain with the data, for example 0 orders of baguettes for almost a year. We solved the issue by meeting with the client and asking for explanations.

* **Modin library**

Adding Modin to increase Pandas performance was not a great idea. We started to have compatibility problems with other libraries and, in the end, we had to remove it.

* **Prophet library**

Prophet library has a bug that affects the matplotlib plots, it took a while to discover why something.plot() was not working.

* **Integrating tableau with TabPy**

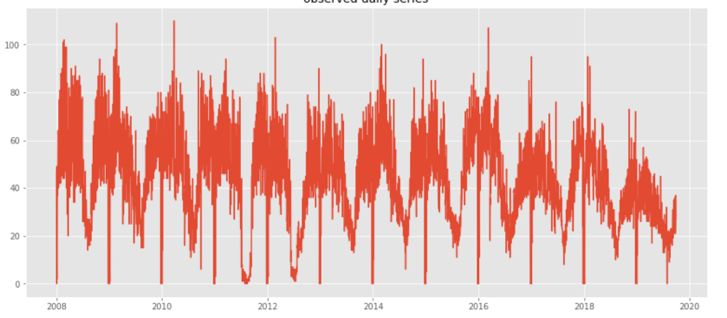
It was also not straightforward, especially with Prophet which, in mac, had another bug that prevented to executing the project from a virtual environment. We solved it by implementing a SARIMAX in tableau, instead of the prophet model.

* **Team coordination**

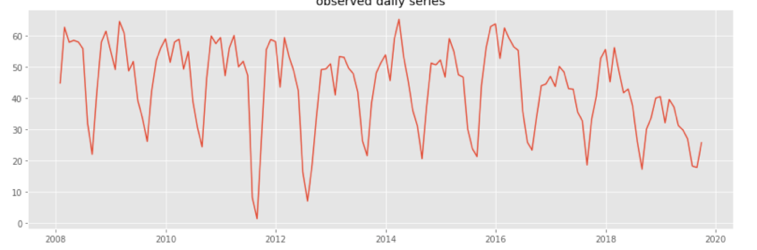
Although were only two, it was hard to coordinate ourselves, and we had some branches incompatibilities that prevented pull requests to be merged.

# Project Results

Without more hesitation, let’s have a look at the predicted time series, the performance of the evaluated models, and unveil the winner model:



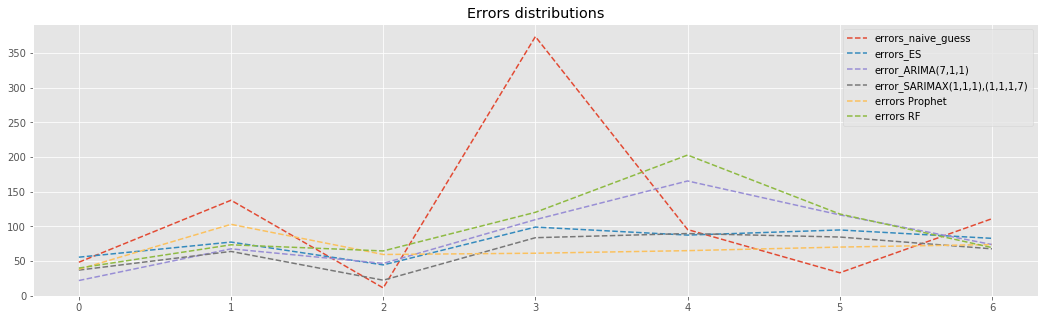
*Daily aggregation*



*Monthly aggregation*

TWO LINES OF OUR SERIES… SEASONALITY, ETC…

Now, let’s have a look at the cross-validation results (mean square errors) for the 7 day forecasting period requested by the client:



Squared ordered units

Forecastin days

Now in table format:

|  |  |  |
| --- | --- | --- |
| **model** | **Mean square error** | **Mean equivalent percentage error** |
| general\_mean | 222.17 | 0.3413 |
| weekday\_means | 174.99 | 0.3029 |
| Naîve\_guess | 86.44 | 0.2129 |
| Holt-Winters | 54.22 | 0.1686 |
| ARIMA | 60.26 | 0.1777 |
| SARIMA | 44.96 | 0.1535 |
| Prophet | 47.16 | 0.1572 |
| Random Forest' | 68.92 | 0.1901 |

As displayed by the results, the winner models where: SARIMA and the Prophet.

Although we would have expected the Random Forest to perform better, it was not great surprise due to the

Interpretación de los errores.

# Conclusions & Improvements

## PROJECT CONCLUSIONS

* **The bakery would benefit** from cleaning the data, and then applying Time Series and Machine Learning models to better manage their workforce.
* **Machine Learning is not always the winner**; the winner model is the one that predicts the best in under any circumstance, and is very dependent on the quality of the input data.
* **Internet was of great help** – stack overflow, medium, etc. We took many ideas and reused code from many sources, (and we always tried keep the reference for the sites that contributed the most to our project).
* **It is not possible to document absolutely everything**. There were many ‘quick-checks’ and ‘forthcomings’ that was not possible to reflect.
* We spent **too much time cleaning the data**, perhaps we should have limited the scope of the products to 5 instead of 10. This would have given us more time to conduct the following improvements, and yield even better forecasting results:

## IMPROVEMENTS

* **Multi variant Random forest & SARIMAX**, with weather, football matches, public holidays, etc.
* Taking into consideration the **yearly seasonality** found in our data, in favour of the weekly seasonality found. Taking into consideration both would probably result in better models.
* Include **LSTMs** networks.
* Separating the series in two: a weekly average and the differences from each date to the mean value, and trying different models to forecast both independently.

## PERSONAL CONCLUSIONS

* Working as a team has been challenging, but rewarding. It was clear that the profile of both supplemented.