Sweet, sweet Data Science

Predicting factory demand of a bakery

By

Miguel Ruiz Nogues

Illán Lois Bermejo

Introduction

The objective of the following project is to forecast the orders that the stores of a bakery chain submit to the factory.

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

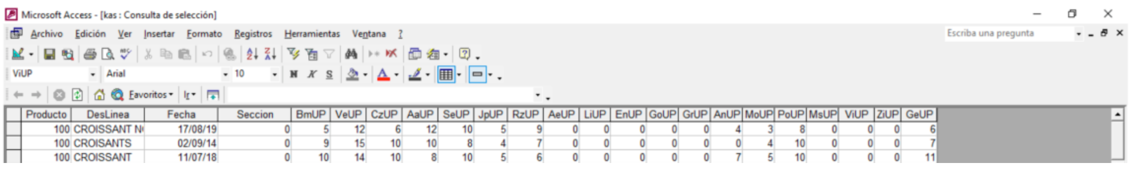
Currently, the factory receives the orders of a given day at 9:00 am of the previous day (in PDF), leaving little merging for errors. In order to schedule the workforce and to stock the necessary ingredients, the rely on expert knowledge, rules of thumbs, and over-resourcing. Although the company stores historical data, they have never utilized it, in fact, the one time they tried to use it, the one IT employee delivered the data in paper.

For this reason, the main goal for our project, is to showcase the client (director of the bakery factory) the power of the data they currently own to leverage their business. We will be doing this by creating a predictive model for one of their products, and 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, cleaning the 10 most interesting products - from clients perspective, forecasting 1 one, and creating a dashboard with all of them.

Raw data

1. <u>*0_original_b2.csv*</u>
   The main dataset used 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 stores ordering that product (as displayed below):


Microsoft Access - [kas : Consulta de selección]
Archivo Edición Ver Insertar Formato Registros Herramientas Ventana ?
Escriba una pregunta
ViUP
Arial
10
Favoritos

Producto	DesLinea	Fecha	Seccion	BmUP	VeUP	CzUP	AaUP	SeUP	JpUP	RzUP	AeUP	LiUP	EnUP	GoUP	GrUP	AnUP	MoUP	PoUP	MsUP	ViUP	ZiUP	GeUP
100	CROISSANT N	17/08/19	0	5	12	6	12	10	5	9	0	0	0	0	0	4	3	8	0	0	0	6
100	CROISANTS	02/09/14	0	9	15	10	10	8	4	7	0	0	0	0	0	0	4	10	0	0	0	7
100	CROISSANT	11/07/18	0	10	14	10	8	10	5	6	0	0	0	0	0	7	5	10	0	0	0	11

   It is important to note that the client provided us a second dataset named *0_original_b2.csv* containing orders prior to 2008, however it was not used due to: age of the data, different stores operating from current stores, etc.

2. <u>*stores_locations.csv*</u>
   Containing geographical information of the stores. The dataset was manually created based on the information of their website, and google maps coordinates.

3. <u>*products.txt*</u>
   Containing the catalogue of products offered by the bakery. The dataset was manually created from pdf catalogues hosted on their website, and it was used to cross-check product descriptions.

4. <u>Other data sources</u>
   Other data sources where collected in order to study correlations with the time series, however due to the lack of time, and high autocorrelation of the time series, they were not utilized:
   a. Local festivities calendar
   b. Local football matches calendar
   c. 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 environments, 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 10 products, and an embedded 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 was created to gain more understanding of the data.
5. Execute 02_data_normalization_and_filtering.ipynb, which was created to filter the 10 products selected by the client.
6. Execute 03_data_cleaning_and_quality_assurance.ipynb, which was created to clean the time series of the selected products, and ensure that 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, 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 around 1500 unique products id, and the relationship between product id and product description was not respected. In other words, under a product id X, there were many products decriptions corresponding to different products e.g. product id X, could have orders for 'baguette', , 'tarta de manzana', 'tarta de queso' , and 'trta qeso', under product description.

  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. (In the baguette case, it was a business decision to stop production).

- **Modin library**
  Adding Modin to increase Pandas performance was a failure. 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.

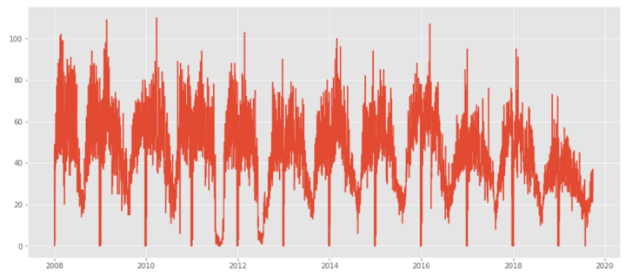
- **Integrating tableau with TabPy**
  Implementing Tabpy was also not straightforward, especially with Prophet which, in MAC, had bug that didn't allow the execution of the project from a virtual environment. We solved the

issue by copying import configurations from the internet, however due to the lack of understanding of what were actually configuring decided to implement SARIMAX instead.

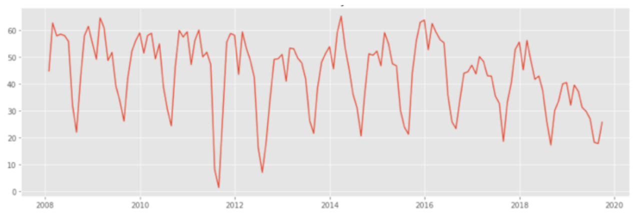
- **Team coordination**
  Although we were only two, it was hard to coordinate ourselves, and we had some branches incompatibilities that prevented pull requests to be merged.

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. First, let's have a look at the time series:



Time series of Palemeras de chocolate - daily aggregation

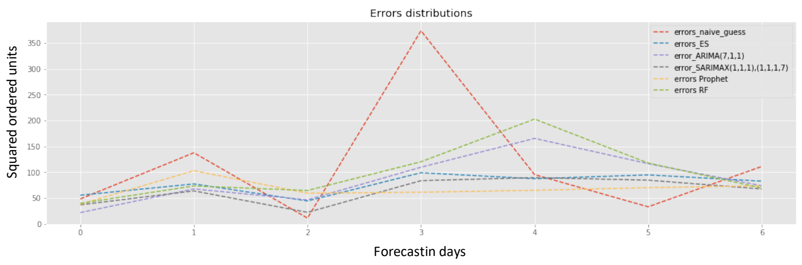


Time series of Palemeras de chocolate - monthly aggregation

From the analysis we conducted, we observed that this were the main features of the time series:

- Strong yearly and weekly seasonality as per ACF and PACF results.
- Stationary as per Dickey-Fuller.
- Without clear trend.
- High variance.

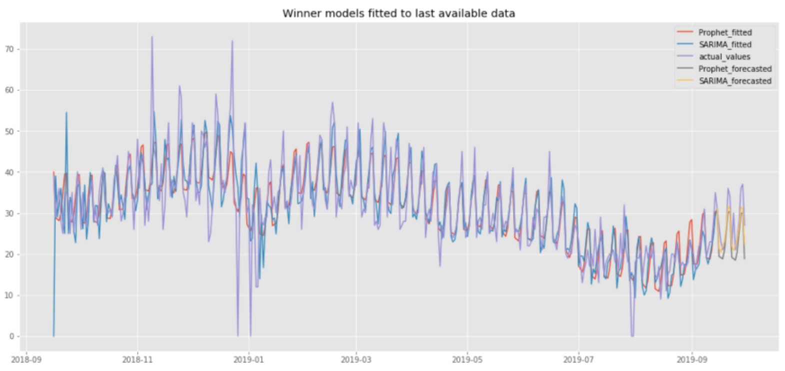
With this analysis in mind, and using cross-validation, we assessed the mean performance of 5 different models against three benchmarks (mean, weighted mean and naïve guess), using the mean square error as the performance metric, due to the amount of 0 value, and high variance of the time series. These were the results:



model	Mean square error [mse]	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	47.08	0.1571
Prophet	47.15	0.1572
Random Forest'	72.5	0.1949

As displayed by the results, the winner models where: SARIMA and the Prophet with an average MSE of 47, and equivalent percentage error of 15% (approximately 6 'palmers de chocolate' of deviation).



Although an equivalent percentage error of 15 is good, it's not great, and can be improved. However, when benchmarked against the current forecasting methods utilized at the bakery factory, it outperforms.

It would have been great to benchmark our model against the real output of the bakery factory but, unfortunately, they don't keep a reliable record of what they dispatch to the stores.

Conclusions & Improvements

Project Conclusions

- **The bakery would grandly benefit** from using our models as they are, however, we believe that investing additional time would yield better results, as we will see in the improvements part.
- **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.
- **Random Forest didn't meet the expectations,** however we believe that it would grandly benefit from external features that help explaining the forecasted time series.
- **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.
- **Quality code standards should be applied at all times**. Leaving it for a refractor workshop is not effective, because as long as the code works, you prioritize other tasks from the backlog.
- **Data quality** seems to be under rated, however is extremely important.
- **A model is never perfect**, it can always be improved, however it's a skill to find the balance between effort and increment of results, for the given purpose of the model.
- 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, etc.
- Taking into consideration the **yearly seasonality** found in our data, in favour of the weekly seasonality. 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 each other.
- **Predictive models don't compete against professional experience,** on the other hand they feed each other creating a synergy.