Nulogy - Retail Store Customer Demand

Evan Falcone

2019-08-22

Nulogy - Retail Store Customer Demand

Nulogy is tasked with building a Machine Learning product for predicting customer demand per SKU at a physical retail store. Below is the iterative approach I would take to building out this product:

1. Identifying stakeholders and product owner(s)

The first step is to identify the stakeholders at play - notably, who on Nulogy's end and who on the retail store's end should be involved in the discussions for setting requirements, building out, taking ownership and maintaining the data product. It is essential that a product owner be identified early on at this stage (be it one person - product manager, a team from one department - logistics team, or an inter-departmental team).

2. Identifying goal of product

Once the right people are looped into the project, it is a matter of brainstorming and identifying requirements for what goal the tool needs to accomplish. For example, I envision the customer demand tool acting as a **decision support** for the retail store's buying team. In other words, the product would return quantities of customer demand per SKU per day or week (i.e. for a given lead time, given as a function of the company's expected in-bound ordering time). Then, the buying team can use their domain expertise to filter through the recommendations for "extreme" ordering quantities and correct them. The idea is the tool supports decision-making by automating 80% of the expected ordering patterns so that the buying team can focus on the 20% of extraneous values. Another stakeholder may however want **strictly-automated decision-making** - stakeholders must first agree on the goal of the product in order to proceed to the next step.

3. Identifying evaluation metric of impact + assessing initial impact

Next comes associating a value and an evaluation metric to said goal - this is usually an estimation at first to determine the priority of the product's development within the organization, usually based on its **impact on an agreed upon business metric.** For example, a customer demand tool for a retail store based on fashion may aim to optimize profit and stock-out by the end of a fashion season (i.e. be as close to sold-out of current-season stock before next season's stock is officially available, while still retaining as much profit as possible). To assess the impact of the product, you must **first define the metric of interest** that it will impact. Considerations for the impact might include how many working hours multiplied by the hourly wages of the buying team would be reduced (minus the cost of developing the customer demand tool) by integrating it into their operations.

4. Identifying Data Product Requirements + Data Requirements

Then, we must identify the product and data requirements the tool will need to incorporate. Take note of the data and information suggested for inclusion by stakeholders (under the assumption that they are the domain experts and intelligence can be derived from their input), as well as complementing the client's suggestions with your own (rooted in a quantitative/operational persepctive).

Examples of what those requirements for a customer demand product might be include what products (what SKUs are in the scope of the project), software systems, hardware systems and data sources. For data sources, here are examples of data I would collect:

- **Transactional data sources**:
- Historical: Price paid before tax, tax amount, price paid inclusive of tax, quantity purchased per SKU per order, total quantity purchased per order, returns per SKU per order, returns per order, CustomerID, discountPercentage, discountAmount, isDiscounted (flag), isEmployeePurhcase (flag).
- **Order data sources**: POs, POitems (per SKU), POitems_quantities, timestamps of when POs were placed, POitemsReceived, POitemsReceived_quantities, timestamps of when POs were received.
- **Stock data sources**: SKU, CurrentWarehouseStock, CurrentWarehouseStockSelling (allowed to be sold), CurrentRetailStock, CurrentRetailStockSelling (i.e. stock on the floor).
- **Product data sources**:
- Product information data: productID, colour, brand, category, department, dateListed (i.e. date at which it was first put out for sale), timestamps of sales, size, flavour.
- **External data sources**: examples of relevant external data sources may include.
- Calendar data: festive dates on the calendar which may have seasonal impact on demand forecasting); can be represented as encoded flags in the database.
- Short-term weather data: hourly, 36h, 7-day and 14-day forecasts for probability of precipitation (POP), humidity, wind.
- Social listening data: Twitter data (raw tweets, hashtags, images), Instagram data (post descriptions, hashtags, images); similar for other social media platforms.

5. Pulling in the data

This task would cover the **merging, pipelining, ETLing**, etc, of the various data sources that need to be combined for the scope of the customer demand data product. This step would likely be owned by the data engineers and/or architecture team(s), though *Data Scientists would be involved to feed the requirements for a "cleaned" abstraction layer of data* to be used in analysis and model development. At this point, these data requirements are far more explicit (like the ones listed in section 4.) and may be accompanied by ERD diagrams, data views and other architecural data designs.

Some elements of step 6. may be completed in this section; notably, the engineering of new data features to be stored in views (in an abstracted data layer) that are engineered from the existing features stored in the database.

6. Feature Engineering

Though some of step 6. may be completed earlier with the help of data engineers and architects, there will still be a need for additional feature engineering both as a result of exploratory data analysis, as well as for use in model development.

Features that I would consider engineering might be the following:

- **Temporal transactional data**:
- Stock increase quantity of the product in related week, ex. stock transfer quantity from distribution center to store.
- Stock return quantity from customers at a specific week and store.
- Weekly sales quantity of SKU at store.

- Hourly sales quantity of related product from 9 am (open) to 22 pm (close).
- Total sales of each weekday of last 4 weeks.
- Total sales of each weekday of last 8 weeks.
- **Temporal Stock data**:
- Maximum stock quantity of related week.
- Minimum stock quantity of related week.
- Average stock quantity of related week
- ** Customer-based transactional data**:
- How many customers bought this product at a specific week.
- **External data**:
- List (maybe store as JSON) of top trending colors on Twitter/Instagram, etc (keywords from Google Vision API image recognition tagging).
- List (maybe store as JSON) of top trending brands on Twitter/Instagram, etc (keywords from Google Vision API image recognition tagging).

7. Build Model

I will leave this section a bit open-ended as it will result from the previous sections how best to approach it, but its goal will be closely tied to the goal set forth in **section 2.** (identifying goal of product). Depending on the product, you will have to determine whether the final output for customer demand will be discrete (number of shirts purchased: discrete, or can do continuous and round, though it introduces rounding error) or continuous (lbs of meat: continuous, if sold at a butcher for example - though a small-time butcher might not need this data product).

8. Evalute Model

Once the model is trained and delivers predictions on a test set, we must evaluate its performance using the evaluation metrics set forth in **section 2.** This will once again be determined by how you frame the problem.

I would frame the customer demand problem as a classification problem where I limit the discrete categorical (ordinal) possible target variable values. This would help take care of issues like curse of dimensionality, though it may introduce other issues such as imbalanced categories in the target variable of our training data.

Common metrics for classification problems would include AUC (Accuracy score) and *Log-loss* score. The trickier part is defining what to test against since there is a temporal notion: you should set a lead time called a "days cover", i.e. have your target variable be the discrete amount of sales expected during the time period it takes you between placing a PO and receiving additional stock in your warehouse + have it be ready to sell.

9. Put Model into Production

This step involves re-convening with the Data Engineers, Machine Learning Engineers and Architecture teams to house your model in a software structure that will allow it to run in real-time without disrupting other systematic operations.

10. Iterating + Maintaining

This step also requires discerning who will own the maintaining of the data product and who will own the iterating. Most often the iterating of the model falls into the hands of the Data Science team in order to improve model performance (with respect to evaluation metrics, sometimes also efficiency though that may fall more under Machine Learning Engineers and Data Engineers).