Forecasting Demand of Perishable Products Using Machine Learning

Raymon van Dinter

ORL-33806
Data Driven Supply Chain Management
Wageningen University and Research

Abstract

Retailers that sell perishable products should manage their supply chain in order to have the least amount of waste or shortages as possible. A retailer in barbecue products provided data to be used during this assignment. This retailer recorded its demand on perishable products for 4 years. A model was created to simulate the operation of common retailers handling with perishable products. An attempt to improve usual demand prediction was sought after using machine learning models, which were fit with features of weather data from the Royal Dutch Meteorological Institute (KNMI).

Keywords: Data Driven Supply Chain Management, Machine Learning, Deep Learning, Regression

1. Introduction

As an assignment for the course ORL-33806, a simulation model was developed to simulate behavior of common perishable retail products. Afterwards, a machine learning estimator was used to created to predict demand, which influences the order-up-to level and order quantity.

Section 2 describes the datasets that were provided and its features. Section 3 describes how the retailer simulation model was set up. Section 4 describes how the dataset was preprocessed and how features were engineered. Section 5 describes the machine learning estimators that were used and their demand prediction scores. Section 6 quantitatively compares the estimators using the retailer simulation model. Section 7 describes the impact of including weather data, and at last, section 8 concludes on all these observations.

2. Provided datasets

Datasets that were provided during this assignment were: i *Data2014-2017DemandWeather.xlsx*, ii *Data2014-2016DemandWeather.xlsx*, and iii *Data2017DemandWeather.xlsx*.

The latter two datasets were split from the first dataset in order to enable users to easily use the dataset with data from 2014 to 2016 as train data to predict the demand of 2017. The datasets were merged demand data from a retailer in barbecue products, and weather data from the same day, retrieved by the KNMI. The datasets consist of four features:

Date Date of sample in *YYYY-mm-dd* format. Sample period is one day.

demand Demand of products during the sample period.

TempTimes10 Average temperature in 0.1 °C during the sample period.

RainfallTimes10 Total rainfall in 0.1 mm during the sample period.

3. Retailer simulation

In order to investigate the operation of a common retailer of perishable products, a quantitative model was developed. This model kept in mind a daily routine:

- At the start of every day, the order-up-to level S will be set by predicting demand D. There are various methods to calculate S, which will be explained later in this section.
- Order quantity Q is determined by subtracting the daily starting inventory I from S.
- During the day, customers buy products. This demand is subtracted FIFO-wise from I. If demand is higher than stock, a shortage is counted.
- Next day starting inventory I is determined by adding Q to I FIFO-wise.

- Waste is computed by counting the expired products, which are products that were in stock for longer than 7 days.
- At the end of the iteration, average shortage and waste are computed.

3.1. Demand input

In this simulation model, two methods of demand D input were implemented. In the first method, D was set straight from the `Demand` feature from one of the datasets. The second method takes a more complex approach; a demand prediction model is trained using one of the datasets, hereafter, the model is able to predict demand over a certain period using weather data, which then sets D.

3.2. Computation of the order-up-to level

Order-up-to level S was computed by using the formula:

$$S = \mu_{R+L}(t) + ss(t) \tag{1}$$

Where R means the daily ordering and L stands for the lead time. $\mu_{R+L}(t)$ is the mean demand over $R + L$ days, and $ss(t)$ is a safety stock that is added to the expected demand. This safety stock was set to half the prediction of $\mu_{R+L}(t)$. Therefore, order up to level is set to $S_t = 1.5 * \mu_{R+L}(t)$.

3.3. Reference score

For each of the datasets, a reference score was computed, which can be seen in table 1. S with $S_t = 1.5 * \mu_{R+L}(t)$, where $\mu_{R+L}(t)$ is the predicted demand over the current day and the day after. The demand of these days is predicted by taking the mean of the 3 preceding weekdays, which means:

$$S_t = 1.5(\frac{\sum_{i=1}^{3} d_{t-7i}}{3} + \frac{\sum_{i=1}^{3} d_{t-7i+1}}{3}) \tag{2}$$

Performance was determined by calculating the Root Mean Squared Error (RMSE), the shortage ratio (also called fill rate) and the waste ratio.

Dataset (years)	RMSE	Fill Rate	Waste Rate
2014 - 2017	7.22	0.32	0.12
2014 - 2016	7.55	0.33	0.13
2017	6.51	0.28	0.07

Table 1: Reference scores of the simulation model

4. Preprocessing the dataset

Machine learning estimators generally use numerical data to create a useful model. The dataset which was provided, still contained categorical data - the `Data` feature - which was converted into numerical data in order to gain performance. This was achieved by converting the date into the day of the week, and then using the `pandas.get_dummies()` function (one-hot encoding). With this conversion, the model is able to differentiate days from each other, which is very useful since people generally buy more barbecue products in the weekends.

After one-hot encoding, the dataset increased from 4 to 11 features, each added feature represents a single day in the week. To add more importance to the weather combined with the day of the week, 2-degree `PolynomialFeatures` were added to the dataset, which increased the dateset to 65 features.

Several of the machine learning models that were used, are sensitive to scaling. Therefore, the `StandardScaler` was used to scale all features. The `StandardScaler` ensures that the mean of every feature is 0 and that variance=1.

5. Prediction models

Demand prediction models that have been used within this research are elaborated in this section.

5.1. Analysis and Choice

In order to forecast demand, several prediction models were created. To achieve relevant predictions with numerical, continuous output, machine learning models used their regression variant. Choosing the right estimator may be a difficult path. Fortunately, sklearn provided a useful schematic to validate the usage of the correct estimators (Choosing the right estimator scikit-learn 0.22.1 documentation, n.d.), which has been used next to the skills gained during the course. Estimators that were researched are:

- Ordinarily Least Squares
- Lasso regression
- Decision tree regression
- Linear support vector regression (SVR)
- Random forest regression
- Multi-layer perceptron regression (MLP)

Linear regression is a very powerful estimator in high-dimensional datasets, and with its L1-regularization,

Lasso could probably be an even more powerful estimator since the dataset has many features gained from the one-hot encoding and 2-degree polynomials.

Furthermore, a decision tree regressor could be higly valuable due to its conditional explainability. Random forest regression was also taken into account due to this benefit.

At last, multi-layer perceptron regression was researched due to it being the main convolutional neural network discussed during the course. When using this model, it was noticeable how powerful it is, with its only big downside being its explainability.

5.2. Parameter settings

In order to efficiently evaluate each machine learning model, the `GridSearchCV` object from the `sklearn` library has been used. `GridSearchCV` allows users to provide an estimator and a range of parameter settings, which optimum values will then be determined.

Table 2 shows every model, its optimal parameter and its scores on train and test data for predicting demand using temperature and precipitation data. Train data is represented by the dataset containing data from 2014 to 2016, while the test dataset contains data from 2017.

The models discussed seem to achieve similar scores, except from the decision tree and random forest regression. Therefore, these models were not further tweaked to be implemented in the simulation model.

6. Quantitative comparison with the simulation model

The linear and MLP regression models were further investigated by scoring them on the retailer simulation model.

Order-up-to level S was calculated by the formula $S_t = ss(t) * \mu_{R+L}(t)$ where $\mu_{R+L}(t)$ is the demand predicted by the estimator of that current day, $D(t)$, and the day after, $D(t+1)$, which means:

$$S_t = ss(t) * (D(t) + D(t+1) \tag{3}$$

Furthermore, $ss(t)$ had a default value of *1.5*, but a value to optimize the scores was sought after. One risky effect could be overfitting, which should be avoided at all cost. The results of this research can be found in table 3

Model	RMSE	Fill Rate	Waste rate
Linear	0	0	0.011586
Lasso	0	0	0.011560
Ridge	0	0	0.011586
Linear SVR	0	0	0.011586
MLP	0	0	0.011540

Table 3: Optimized models and their scores when used as input data to calculate S. $ss(t) = 1.5$

Model	RMSE	Fill Rate	Waste rate
Linear	0	0	0.000658
Lasso	0	0	0.000649
Ridge	0	0	0.000658
Linear SVR	0	0	0.000658
MLP	0	0	0.000636

Table 4: Optimized models and their scores when used as input data to calculate S. $ss(t) = 1.1$

Model	Parameter values	Score (train, test)	RMSE (train, test)
Linear regression	{'normalize': False}	(1,1)	(0,0)
Lasso regression	{'alpha': 0.0001}	(1,1)	(0,0)
Ridge regression	{'alpha': 1e-07}	(1,1)	(0,0)
Linear support vector regression	{'C': 100}	(1,1)	(0,0)
Decision tree regression	{'criterion': 'mse', 'max_depth': None, 'random_state': 0}	(1, 0.99)	(0, 0.228)
Random forest regression	{'max_depth': 8, 'max_features': 'auto', 'n_estimators': 500, 'random_state': 0}	(1,1)	(0.196,0.189)
Multi-layer perceptron regression	{'alpha': 1e-06, 'hidden_layer_sizes': [10], 'max_iter': 1000000, 'solver': 'lbfgs, 'random_state': 0'}	(1,1)	(0,0)

Table 2: Model parameters and their scores on predicting demand using temperature and precipitation data.

After carefully investigating the values of the waste rate, $ss(t) = 1.1$ was the optimal value of the safety stock. Furthermore, it can be observed that Lasso and MLP had the lowest waste rate.

7. Impact of not including weather data

In order to review the importance of weather data, the models were retrained and preprocessed. However, this time when preprocessing, weather features were dropped out of the dataset. Table 5 shows the scores of each model. It is clear that differences for many of the models are not visible. However, Lasso and MLP show a negative difference.

Model	RMSE	Fill Rate	Waste rate
Linear	0	0	0.011586
Lasso	0	0	0.011575
Ridge	0	0	0.011586
Linear SVR	0	0	0.011586
MLP	0	0	0.011579

Table 5: Optimized models and their scores when used as input data to calculate S, this time trained without weather data. $ss(t) = 1.5$

8. Conclusions and Recommendations

From the results observed in tables 3 to 5, it can be concluded that Lasso and MLP are the best regression models to determine D. Furthermore, $ss(t) = 1.1$ and including weather data provides the best scores. If a model must be explainable, Lasso regression would be the estimator of choice. Otherwise, MLP would be chosen due to its higher scores. Lasso's L1-regularization means that many of its features are set to zero, meaning it can be written down like:

$$\begin{aligned} D_{predicted} = 2.22 &+ 0.407x_0 - 0.502x_1 + 0.003x_9 \\ &+ 1.100x_{11} + 0.992x_{12} + 1.088x_{13} \\ &+ 1.326x_{14} + 1.805x_{15} + 2.198x_{16} \\ &+ 1.375x_{17} - 0.313x_{19} - 0.311x_{20} \\ &- 0.313x_{21} - 0.318x_{22} - 0.317x_{23} \\ &- 0.315x_{24} - 0.311x_{25} + 0.214x_{26} \\ &+ 0.174x_{27} + 0.175x_{28} + 0.263x_{29} \\ &+ 0.229x_{30} + 0.177x_{31} + 0.218x_{32} \end{aligned} \tag{4}$$

Here, every x represents a feature column. Another 30 features are set to 0, which means that some of the features were unimportant.

It can be observed that with using either of these simulation methods, no shortage is occurring anymore, and waste is greatly reduced when comparing with the reference scores from table 1.

When starting to predict demand, using $ss(t) = 1.5$ should be a safe start and only start reducing $ss(t)$ after a certain time, if comfortable.

References

Muller, A. C., & Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists (1st ed.). Culemborg, Netherlands: Van Duuren Media.

Choosing the right estimator scikit-learn 0.22.1 documentation. (n.d.). Retrieved February 1, 2020, from `https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html`