Capstone project summary

Predicting the Write off amount of a product group

Oszkár Egervári
CEU Business Analytics
2021-2022

Project Description

My Capstone project task was to predict the write off amount of a specific product group for my client, a large retailer in Hungary. The goals of my project were to analyze the available data, create predictions for the write off quantities for all their stores in Hungary, and based on that, create a schedule for the future dates of write off removals.
Prior to my project, my client didn't have a prediction model implemented for this purpose. The schedule was planned in an ad-hoc manner, resulting in under- and overestimating the amount of waste. The goal of my project was to save cost on waste removal via accurate prediction.

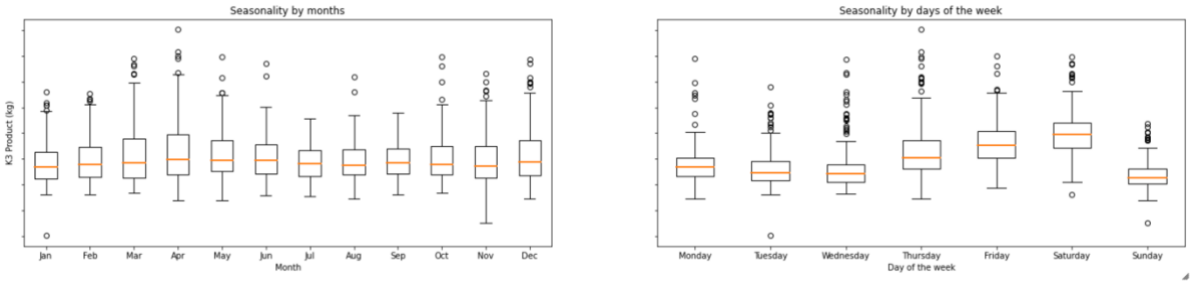
Data Preparation

My work started with gathering the necessary data. I used past years write off amounts for all the store and aggregated it to daily amounts per store. Another table contained the weight information of each product. By joining the two tables, I acquired the outcome variable, which I needed to predict for the next year.

Then, from my client's database I acquired the transactions data for the same products and period, and similarly to the write offs, I converted the transactions to weight. Even though, time series forecasting can be done without explanatory variables, I thought it to be advantageous to analyze the transaction data and explore the potential relationship between transactions and write offs of the same products, before going forward with prediction.

Exploratory Data Analysis

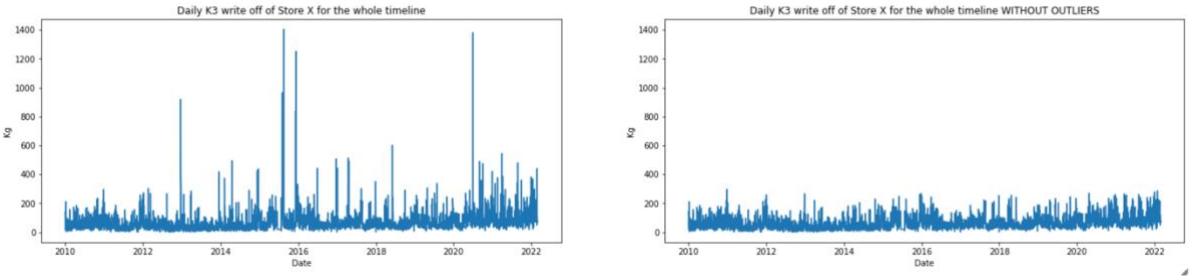
Before creating models for the predictions, it was essential to understand the data, see if there are outliers, seasonality, trend, or any patterns. Since, as I mentioned earlier, I was curious about the connection between waste and transaction of the same products, I did the EDA for both write off and transaction data.
I found out, that although the two are correlated, both possessing multiple levels of seasonality, transaction amount tend to change more during a week, while write off changes more from month to month. This surprised me, as I expected the transactions to follow a yearly seasonality to a much larger extent. Figure 1. displays the two levels of seasonality of transactions:



1. Figure: Monthly and Daily levels of seasonality of a certain product group

Outlier Detection

The historical write off data contained numerous outliers[1]. After consulting with my client, we decided that it's best to remove them, as some of them are errors, while others are not relevant, according to them. Figure 2. plot shows the write offs for the whole timeline with and without outliers:



2. Figure: Write off with and without outliers

This resulted in a lower average and standard deviation for the write offs. I fitted and validated the models – out of which I chose the best performing one for my final prediction – on both data with and without outliers. Although their rank didn't change, their performance[2] improved significantly.

	Without outliers	With outliers
count	4220.00	4220.00
mean	61.30	67.05
std	44.61	70.92
min	0.60	0.60
25%	30.14	30.14
50%	50.62	50.62
75%	77.82	81.19
max	296.34	1405.00

1. Table: Stats with and without outliers

Prediction Models

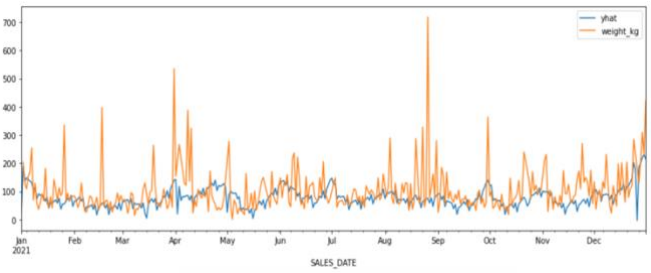
My plan was to start out with the more conventional methods, then try my luck with newer technology. First, I created and tested numerous different ARIMA[3] models. I set their parameters according to

[1] To detect outliers, I used *Prophet* prediction for the same period, and labeled the observation outside of the 99% prediction interval outliers.

[2] I used RMSE to evaluate the model performance

[3] ARIMA stand for Autoregressive Integrated Moving Average. The *AR* part can be determined through analyzing the partial autocorrelation, which is the number of lag observations in the model. The *MA* part can be

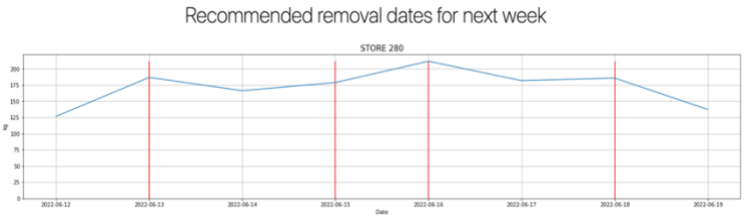
my analysis on the autocorrelation and partial autocorrelation of the data, but ultimately, I used pmdarima's[4] auto ARIMA function, which uses machine learning to determine the optimal order of an ARIMA model.
Out of the five models I fitted on the test set, I chose the best performing one for validation. Then I plotted the predicted values against the actual values of the validation set, and I realized, that the best ARIMA model was unable to follow the in-week seasonality of the data.
Next method I tried was *Prophet*[5]. I created four models with slightly different parameters. I tested their performance similarly, as I did with ARIMA models, and selected the best model for validation. The model achieved similar performance on the validation set, as ARIMA, however this time, it was able to follow the in-week seasonality, displayed on Figure 3.



3. Figure: Prophet predictions and actual values

Results

Based on its performance, I selected the best performing *Prophet* model to forecast the write offs for the next year, for all stores. I repeated the same procedure as earlier, just on a larger scale. Based on the predictions, I created an interactive dashboard, where my client can see the upcoming recommended removal dates for a selected store (seen on figure 4), as well as a plot displaying the predictions for the next year, and a table, which can be adjusted to check a store's predicted write off amount, projected removal dates, and a cumulative sum of write off since the latest projected removal.



4. Figure: Write off removal schedule

Conclusion

For the best possible result, I recommended my client a couple of things, that could improve on the model, and consequently result in the most saved cost. However, I believe, that even if my client doesn't find making these changes viable, relying on the projected schedule would still save them money.

determined through analyzing the autocorrelation. The *I* is the use of differencing of raw observations to make the time series stationary.
[4] A statistical library in Python.
[5] Prophet is Facebook's procedure for forecasting time series

During my project I got familiar with a handful of new tools, like Databricks, PySpark, which was a new language to me, and of course, forecasting time series. I was also able to practice my Python and SQL skills, as well as getting to know libraries like *statsmodels* and *Prophet.*
This was also the first time I was able to work with big data, rows in the tens and hundreds of millions, which came with its own set of challenges. One lesson I learned is, that if a method fails to work for whatever reason, it might be time to move on, and stick to the schedule, rather than run around in circles. Another important lesson was, that proper communication is key to a project like this.