A journey into Supermarket macrospace

By Laura Varsandan

City University

INM430 – Principles of Data Science

# Introduction

## 1.1 Analysis Domain

The supermarket industry has an abundance of data which could be used to provide guidance towards decision-making. One such key decision relates to ‘macro-space’: how much space should be allocated to each section in each store. Examples of sections could be Grocery, Produce, Cleaning etc. The ‘micro-space’ then deals with how much space should be allocated to which products within a section. The two different decisions need different approaches. The emphasis of ‘macro-space’ is comparing how sections perform relative to each other, whereas the ‘micro-space focuses on how products within a section perform relative to each other. I came across these terms and concepts during my professional experience at Sainsbury’s, a UK supermarket store, but the concept is used at an industry level (Retail Acumen, 2010).

The key question for this analysis is therefore which sections a supermarket chain needs to increase and which sections it needs to decrease to achieve its sales maximization objective.

## 1.2 Data Sources

In order to develop a solution for the ‘macro-space’ challenge, I am going to use data from Corporacion Favorita, an Ecuador supermarket chain (Kaggle, 2017). They have provided the following data for a Kaggle competition:

* Daily volume of sales by product and store since January 2013 to August 2017
* Daily volume of transactions by store
* Daily prices of oil
* Holiday events
* Store metadata
* Product Metadata.

## 1.3 Key Assumptions

The key limitation of this dataset is that it does not contain how much space each product or section occupies in each store. Therefore, I have decided to use the number of unique products ranged in each section-store combination as a proxy for the size of the section. This important aspect will be further discussed in section 7.

Since the data does not include any profit information, but only unit sales volumes, I will assume that the chain wants to maximize unit sales. This is not an uncommon objective as a lot of supermarket chains pay close attention to market share, particularly in the UK (BBC, 2017).

Since it can be operationally expensive to change store layout, I will assume that this decision is only made once, during the current month for the following month. Since this may be a centralised or a decentralised decision, I will design the analysis in such a way that the results will be usable at all levels, by providing results by store and section, which can then be aggregated up.

The results will not include performance forecasts of new introductions of products in certain stores. The question of how well a new product is going to perform in a store is a separate analytical question which is outside the scope of this analysis.

## Main Objective

The main objective of this analysis is to provide a tool for decision makers that suggests how many products they should allocate to a particular section (in our case ‘family’) in a particular store for the following month. Since July 2017 is the latest month with complete data, that will be used as our target month for the suggestions.

## Analysis Strategy

To achieve our objective, the first step will be to predict sales-per-unit for July 2017 at a day-store-class level, which will then be summed to sales-per-unit at a monthly level for each store-class. Predicting sales-per-unit at the intermediary level between ‘family’ (i.e. section) and product has the benefit of taking into account that different classes within sections might perform really good or really bad. Additionally, it takes the focus away from product-level, which is a ‘micro-space’ decision and should not be dealt with in this analysis. Predicting sales at a day level allows us to consider that certain classes might perform well on certain days and it also adds more points of analysis to train our model compared to doing a forecast at a monthly level. Thirdly, by forecasting at a store level, we can firstly capture better the way classes behave in particular stores.

Providing an output at a store level allows decentralised decision making, where store managers can increase their sales by adding more products to the high performing sections in their store.

A base algorithm will be created and several more advanced algorithms for predicting continuous variables will be compared to it.

The second step will be to use the monthly predictions for July at store-class level in an optimization algorithm that will identify how many unique products should be ranged in each store-class combination to maximize sales for that store.

The last step will be to add the results of the optimization into an interactive visualisation tool designed for the decision-makers in the supermarket company.

# Analysis Software

I used three software tools for this analysis. Python via jupyter notebooks for reading data and manipulation, because Python facilitates relatively easy manipulation of large data sets (Stack Exchange, 2015). I used R for the predictive model (Appendix A) and Python for the optimization model, due to the ease of use.

I used PowerBI as a visualization tool of the end results. Some of the barriers identified for adoption of predictive analytics by business users is the ability to integrate with existing systems and the need for training (Schoenherr & Speier-Pero, 2015). With PowerBI, these two issues are solvable as PowerBI is integrated to Office 365 accounts, making sharing of dashboards easy. Its visualisations are intuitive for users, with limited need for training, an example of which will be shown in Section 6. This step illustrates how the results can be acted upon to add business value.

# Data Manipulation

## 3.1 Loading the data

The data was loaded using Python’s ‘pandas’ package, so that the output is shown as a dataframe [Figure 1 The first 5 rows from the ‘train’ dataset as downloaded from (Kaggle, 2017)Figure 1].

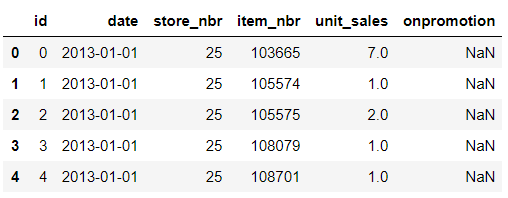


Figure 1 The first 5 rows from the ‘train’ dataset as downloaded from (Kaggle, 2017)

The following table shows the size of raw data, as loaded from Kaggle(2017):

Table 1 Size of original datasets

|  |  |  |
| --- | --- | --- |
| Dataset Name | Number of Rows | Number of Columns |
| holidays\_events | 350 | 6 |
| Items | 4,100 | 4 |
| Oil | 1,218 | 2 |
| Stores | 54 | 5 |
| train | 125,497,040 | 6 |
| transactions | 83,488 | 3 |

## 3.2 Transforming the data

I reduced the size of the ‘train’ dataset from day-store-product level to day-store-class level, merging the ‘train’ and ‘items’ datasets (Figure 3). The only missing values were for the ‘onpromotion’ column, which were missing systematically (Newman, 2014) between January 2013 and March 2014 (Figure 2).

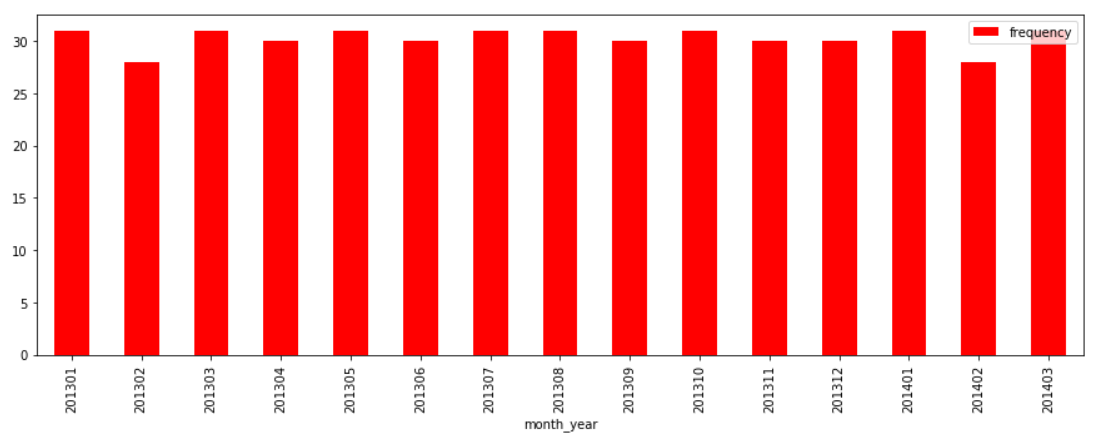


Figure 2 Number of Days with Missing Values

Since the ‘onpromotion’ feature might be one of our features, it would be difficult to accurately impute for such an extended period, which might negatively distort the predictions. Therefore, I only used data from April 2014 to August 2017, which is still a significant amount of data (Figure 4).

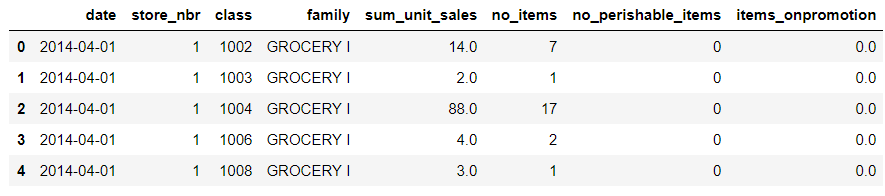


Figure 3 Train Dataset at Day-Store-Class level

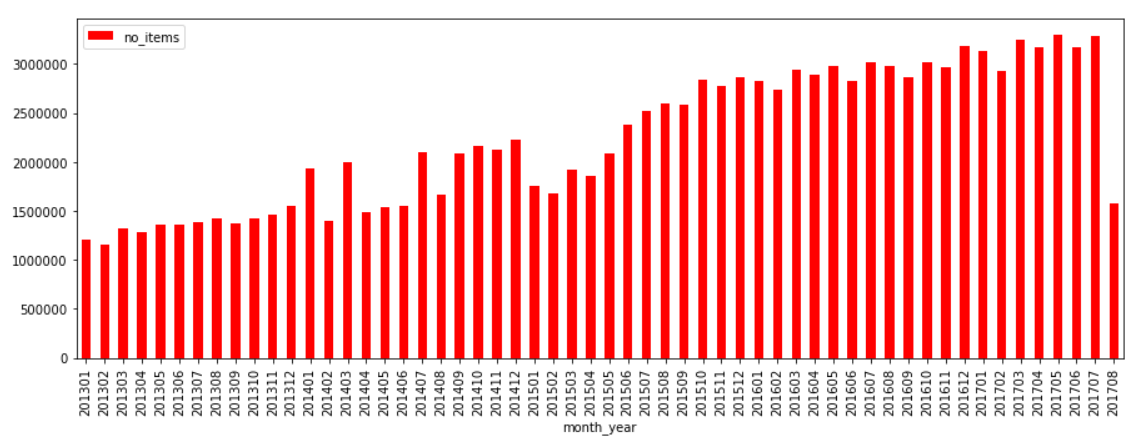


Figure 4 Number of Items ranged per Month

The new dataset had 12,794,614 rows, which became easier to work with for merging it with the remaining datasets and for adding features.

## 3.3 Derived Features

The first feature I built was the target variable, by dividing the total unit sales for each day-store-class combination to the number of unique items ranged, to create a sales-per-unit performance metric which is normalised so that classes with different numbers of products ranged are comparable.

Although the cluster and type for each store are given, their meaning is not explained on the Kaggle website. I therefore decided to create other meaningful features for describing stores:

* Number of products ranged per store per year (Figure 5)
* Number of transactions they receive each year (Figure 6)
* what percentage of transactions occur during the weekdays (versus weekend) - could indicate whether stores are being used as top-up shopping during the week or weekly shopping during the weekend. (Figure 7)
* their city coordinates - there might be regional shopping variations and this will help approximate how close stores are between each other (Google, 2017)(Figure 8).

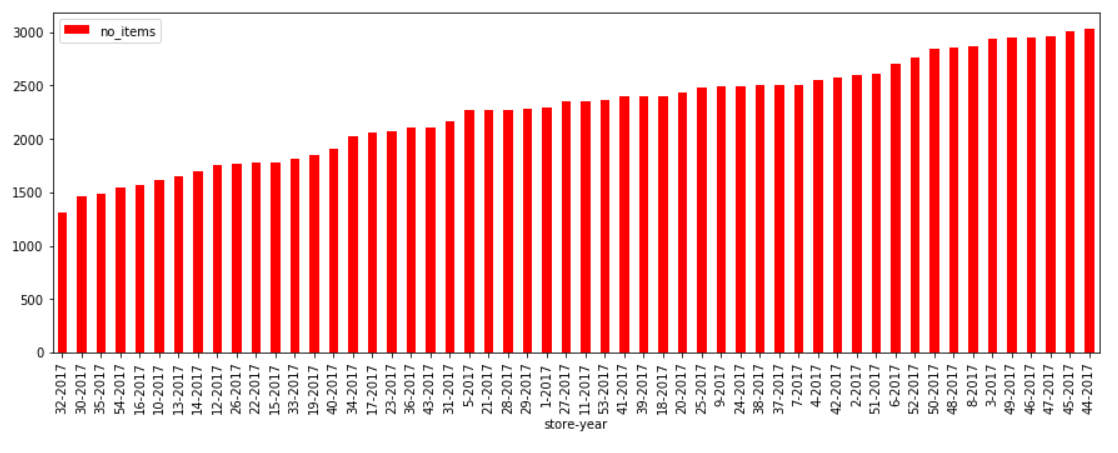


Figure 5 Number of Products Ranged Per Store-Year

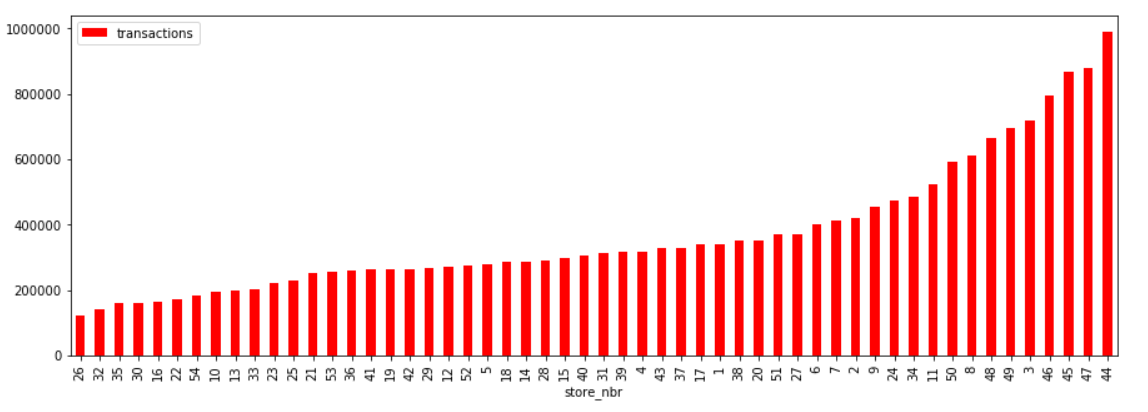


Figure 6 Number of Transactions for 2017 by Store

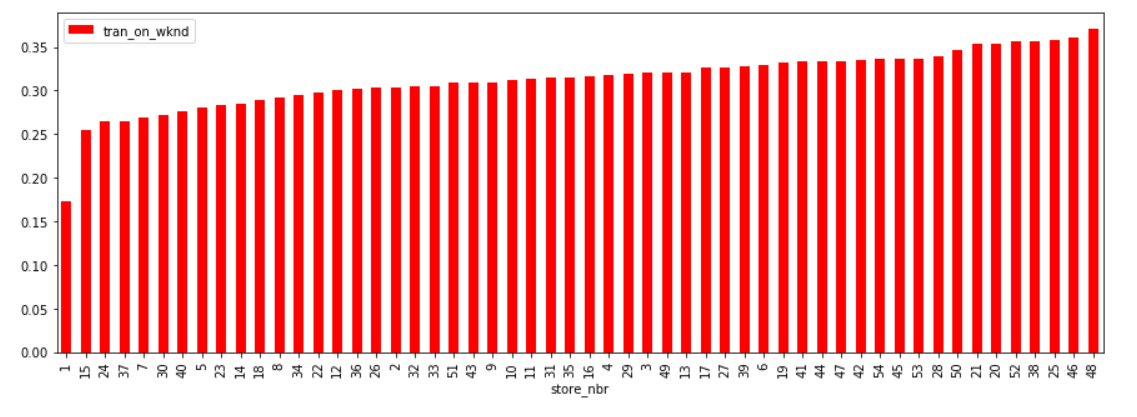


Figure 7 Percentage of Transactions happening on a Weekend

Figure 8 Latitude and Longitude of the cities where stores are located

To reflect shopping behaviour around important events, I also built a wage factor, holiday factor (Figure 9) and an earthquake factor for the April 2016 earthquake (Kaggle, 2017).

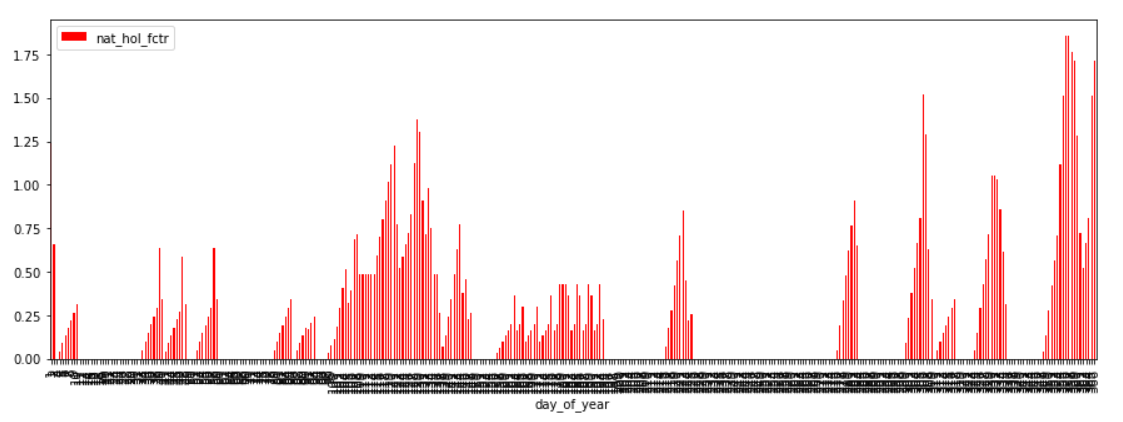


Figure 9 National Holiday factor averages for each day of year

The daily oil price is not likely to have an immediate impact on the sales (Figure 10). Hence, I built a monthly oil price average lagged by one month (Figure 11)

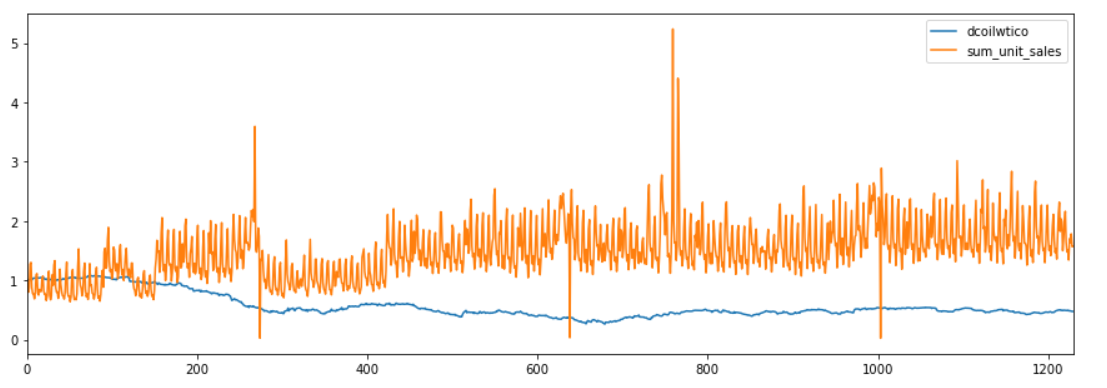


Figure 10 Daily oil price and sales fluctuations based on April 2014 as index.

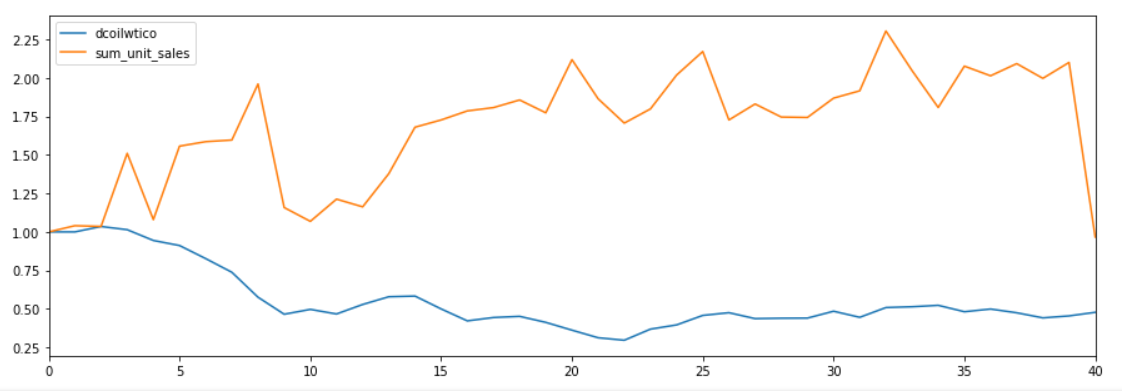


Figure 11 Monthly 1-month lag average oil price and sales fluctuations based on April 2014 as index.

The other categorical features I built were the year, month, day of week, week day indicators and day of month to better capture time patterns. I have also derived a percentage of items on promotions for each class.

# Predictive Model Building

I split the data set into a training and a testing dataset to evaluate the models on unseen data. Since the latest month with full data was July 2017, I decided to use that month as the test set and data up to May 2017 as the train set. I have excluded June 2017 because if this exercise would happen in real life, it will probably happen in June 2017, when the latest month with full data will be May 2017.

I created a base model against which I can compare the more computationally complex models. The base model is an average of our target variable, sales-per-unit, at a year-class-store level. I eliminated the store-classes combinations for which the predictions are null. These are new introductions of classes in certain stores, which are outside the scope of this analysis.

The evaluation of models on the test set was based on two metrics: the percentage of variance explained(VE) and the root mean squared error (RMSE). The former will give us an idea of how much more room for improvement there is, whilst the latter will indicate how far the point-wise predictions are from the actual values

Table 2 Predictive Model Comparison Metrics

|  |  |  |
| --- | --- | --- |
| Model Description | Variance Explained | RMSE |
| Base Model | 55.93% | 7.9191 |
| Simple Moving Average (24% of test data) | 95.5% | 1.206 |
| Regression tree on family | 5.37% | 11.3927 |
| Regression tree on family and store features | 7.43% | 11.28 |
| Regression tree on family, store transactions and month | 7.41% | 11.35 |
| Regression tree on family, transactions, transactions on weekend month and national holiday factor | 7.44% | 11.31 |

The first model was a simple moving average model which is often used for time series forecasting. I created a forecast for each store-class combination (17,172 combinations). Due to computational limitations of my machine, it was not able predict for the whole test set. However, in 9 hours, 24% of the test set had predictions with a RMSE of 1.206 and VE of 95.5%. These are very promising results which could be further refined with different time series algorithms if the right computational resources are available. Since these were not available to me, I have decided to pursue less computationally-heavy algorithms.

The second algorithm was a regression tree, as it can be used on both numerical and categorical variables and it doesn’t make as many assumptions as linear regression models do. I chose a simple tree rather than Random Forest as the Random Forest was too computationally heavy for my machine. Since R’s tree function can only deal with categorical variables that have maximum 32 levels, I merged the Grocery I and Grocery II families, since Grocery II only had one class. I applied a log transformation on the target variable to even out the variance when predicting (Figure 12). I did not remove the outliers as they are important for our analysis, particularly for the high performing classes.

I then tried models with different variable combinations (Table 2).

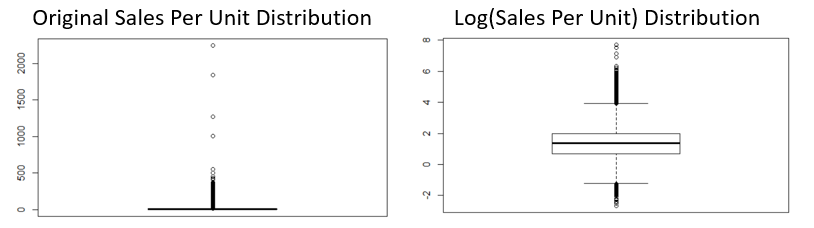


Figure 12 Sales-Per-Unit Distribution before and after log transformation.

None of the regression trees came close to the base model predictions, so I decided to use a regression tree on the residuals from the base model predictions (Appendix B). I had to apply the base model on the train dataset as well and remove 21% of the training data-points since the base model did not have predictions of them.

Table 3 Predictive Models on Base Model Residuals

|  |  |  |
| --- | --- | --- |
| Model Description | Variance Explained | RMSE |
| Base Model | 55.93% | 7.9191 |
| Regression tree on base model residuals with month and day of week | 56.4% | 7.8810 |
| Regression tree on month and weekday indicator | 56.4% | 7.885 |
| Regression tree on family, month, day of week, wage and holiday factor | 56.4% | 7.885 |
| Regression tree on base model residuals with month, day of week, national holiday factor and percent on promotion | 56.71% | 7.866 |

The new target variable was the log transformation of residuals, but since they were negative, I applied a constant to make them positive so that the log transformation could happen. After several model iterations with different variable combinations, the highest variance explained was 56.71% and the lowest RMSE was 7.866. This regression tree used the month, day of week, national holiday factor and percentage of items on promotion as the input variables (Figure 13 ) (Appendix C)

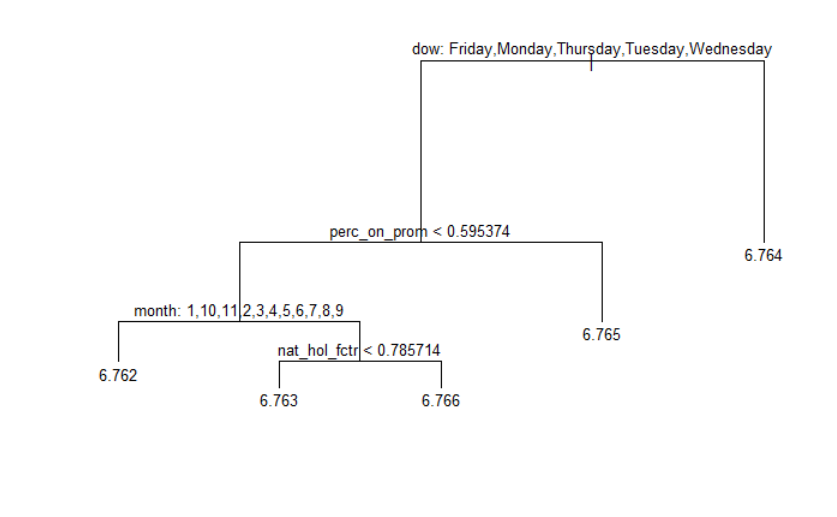


Figure 13 Final Predictive Regression Tree Model

At this point, I was no longer able to detect meaningful patterns that would suggest any potential improvements to the model, so I decided to use this model’s predictions as inputs to the optimization phase.

# Optimization

After summing the sales-per-unit predictions for each store-class to get the monthly sales-per-unit for July 2017, I used a linear optimization algorithm as I defined the optimization problem as a linear function. The decision variables were how many unique products we should range for each store-class combination in July 2017 so that we can maximize our objective function: the total sales-per-unit for each store. The constraints were that the number of products ranged for each store-class had to be non-negative and lower or equal to the maximum of unique products within a class. Additionally, the sum of total unique products in a store could not be higher than the store’s capacity, which I derived by finding the maximum number of unique products ranged for each store-year (Figure 14). The optimization ran in a loop for each store, using code concepts introduced by Slavitt(2014) and Python Hosted (n.d.). (Appendix D – Optimization Python Code.

# Final Results

To understand the final results, I used the unique products ranged in May 2017 for each store-class for comparison to the suggested numbers for July 2017. For visualizing the results, I have created a data model in Power BI (Appendix E), which allows users to look at data at different store levels via filters or Family/Class levels via drill-downs. For example, the store manager of store 8 can see (Figure 15) that he/she should perhaps decrease the size of the Grocery I, Cleaning and Personal Care sections and increase the size of the Beverages, Dairy, Meats and Produce Sections if he/she wants to maximize the number of units sold. He/she could also drill down into the Beverages section to see exactly which classes would benefit from an increase (Figure 16) e.g. 1124.

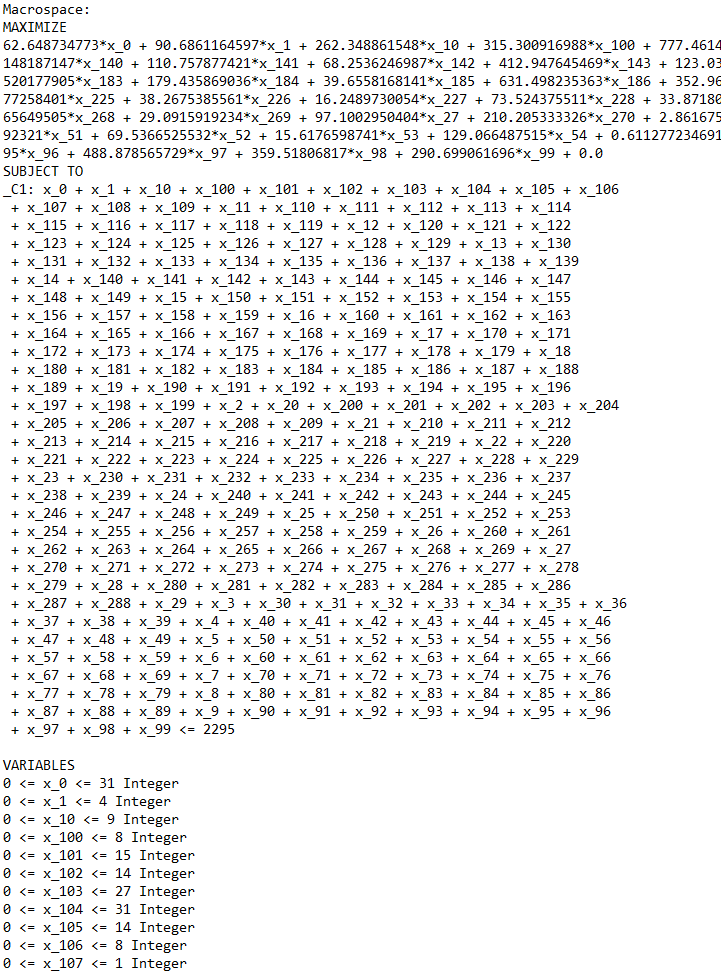


Figure 14 Optimisation Problem Description in Python's Pulp Package

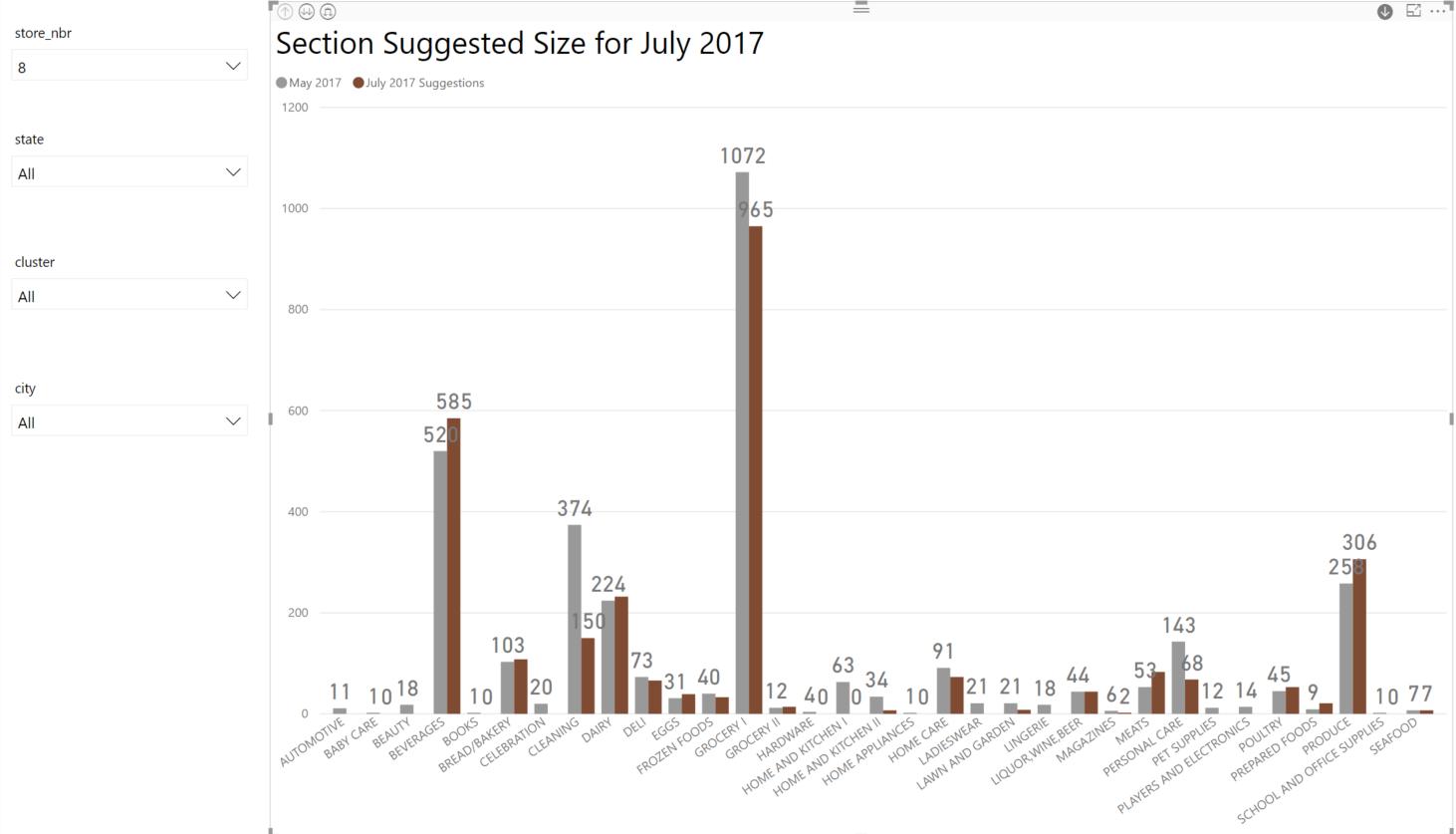


Figure 15 Power BI Visualisation of Results for Store 8 at Family/Section level

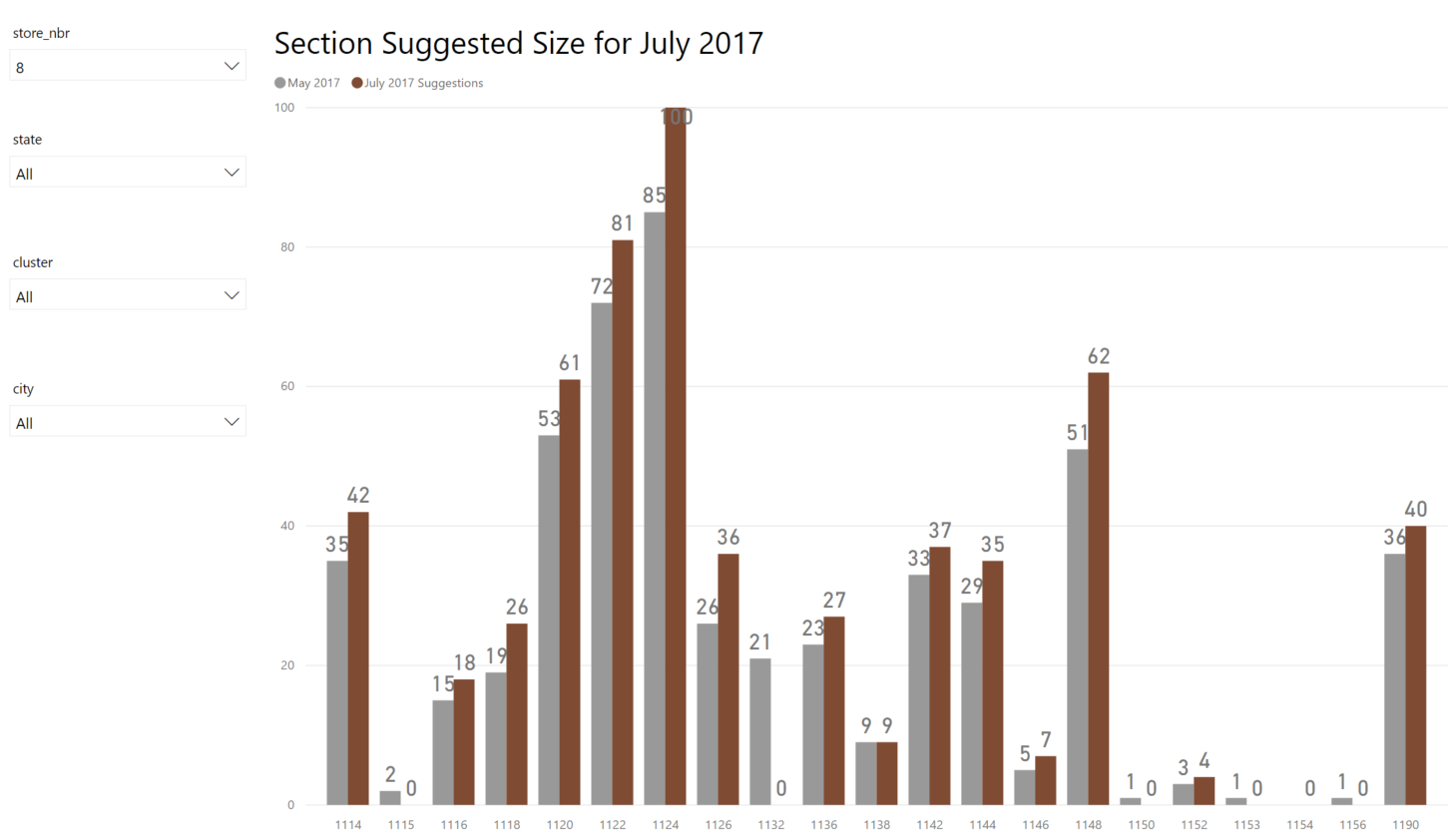


Figure 16 Power BI Visualisation of results at Class level for the Beverages section and Store 8

# Reflections

The main objective of offering decision-makers a tool that would provide guidance towards how many products they should range in each section has largely been met. However, the approach described above has a few limitations. Firstly, it does not consider that in each store there might be a minimum range from each section that should be ranged. As a result, a lot of the small sections like Baby Care or Beauty do not get any recommended products for July 2017. Such extreme scenarios should be interpreted by the decision-makers so that they can apply their domain expertise to further refine the suggested results. Alternatively, a separate analysis should be done to determine what constitutes a minimum range in each section-store and that should be added as a constraint to the optimisation algorithm.

A second limitation would be the prediction accuracy of the forecasted sales-per-unit for July 2017. The variance explained of 56.7% still suggests there is room for improvement and under the appropriate time and computational resources, a better prediction model would ensure better optimization results and therefore it would increase trust in the data.

Thirdly, only using one month for testing purposes is risky as the results for the other months might be different. Therefore, the predictions from this approach would have to be closely monitored every month and fine-tuned. Alternatively, with better resources, a better model could be built that could be cross validated on several months. This would become more robust as we progress to the future, since we would have more data points.

Lastly, with more information such as the space occupied by products, profit, product attributes, store volume capacity, this model could be improved so that it better answers the challenge of utilizing space.

# Conclusion

The aim of this report was to describe an approach to achieve a better space allocation in supermarkets by using open source data from Kaggle. Given the data limitations, the main objective has been refined to provide suggestions to the number of unique items that should be ranged in a section-store combination so that the volume of total units sold for July 2017 is minimized. The first phase of the analysis consisted of developing a predictive algorithm using a base model and a regression tree to predict the residuals from the base model for estimating sales-per-unit at each day-store-class combination for July 2017. The second phase of the algorithm was to optimize how many unique units or products should be allocated for each store-class combination so that the total sales-per-unit is maximized. The last step was to compare the suggested results for July 2017 with the actual levels in May 2017 and suggest ways in which store managers could act to increase or decrease certain sections to maximize the volume of sales.

# References

BBC. (2017, August 22). *Lidl tops Waitrose to become UK's seventh biggest grocer*. Retrieved from www.bbc.co.uk: http://www.bbc.co.uk/news/business-41011259

Google. (2017). *Google Maps*. Retrieved from https://www.google.co.uk/maps/place/0%C2%B010'26.4%22S+78%C2%B028'11.5%22W/@-0.2022042,-78.6098728,12.51z/data=!4m5!3m4!1s0x0:0x0!8m2!3d-0.173986!4d-78.469864

Kaggle. (2017). *Corporacion Favorita Grocery Sales Forecasting*. Retrieved from www.Kaggle.com: https://www.kaggle.com/c/favorita-grocery-sales-forecasting

Newman, D. A. (2014). Missing Data: Five Practical Guidelines. *Operational Research Methods, 17*(4), 372-411.

Python Hosted. (n.d.). *A Blending Problem*. Retrieved from pythonhosted.org: https://pythonhosted.org/PuLP/CaseStudies/a\_blending\_problem.html

Retail Acumen. (2010, 10 6). *Macro Space Planning, Micro Space Planning, Visual Merchandising and Assortment Planning!* Retrieved from retailacumen.wordpress.com: https://retailacumen.wordpress.com/2010/10/06/macro-space-planning-micro-space-planning-visual-merchandising-and-assortment-planning/

Schoenherr, T., & Speier-Pero, C. (2015). Data Science, Predictive Analytics, and Big Data in Supply Chain Management: Current State and Future Potential. *Journal of Business Logistics, 36*(1), 120-132.

Slavitt, I. (2014, June 20). *Using Python for linear optimization*. Retrieved from isaacslavitt.com: http://isaacslavitt.com/2014/06/20/linear-optimization-in-python/

Stack Exchange. (2015, August 27). *Python vs R for machine learning*. Retrieved from datascience.stackexchange.com: https://datascience.stackexchange.com/questions/326/python-vs-r-for-machine-learning

# Appendices

## Appendix A - Packages

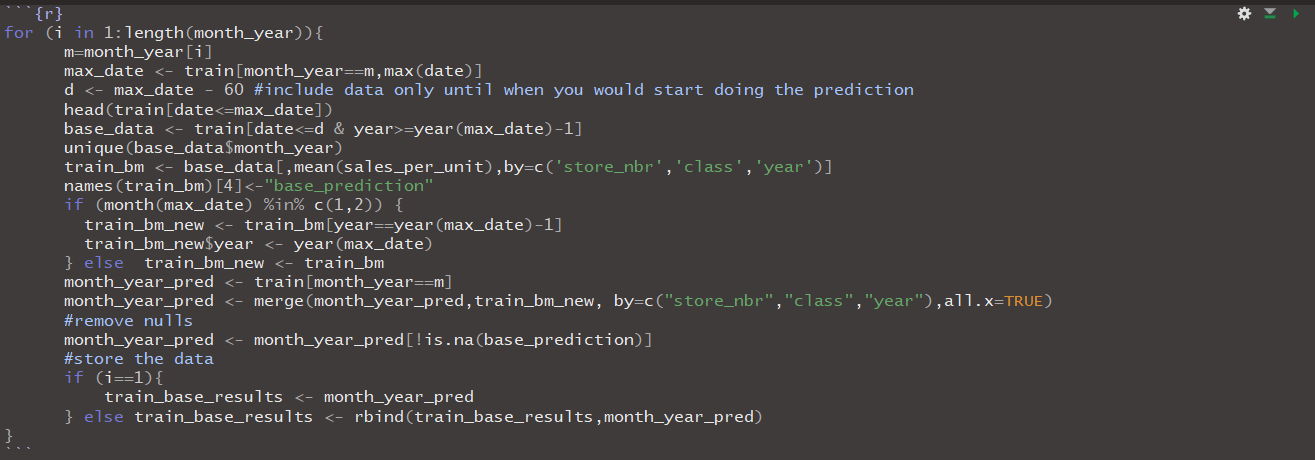
Packages used in Python:

* Pandas
* Numpy
* Matplotlib
* Pulp

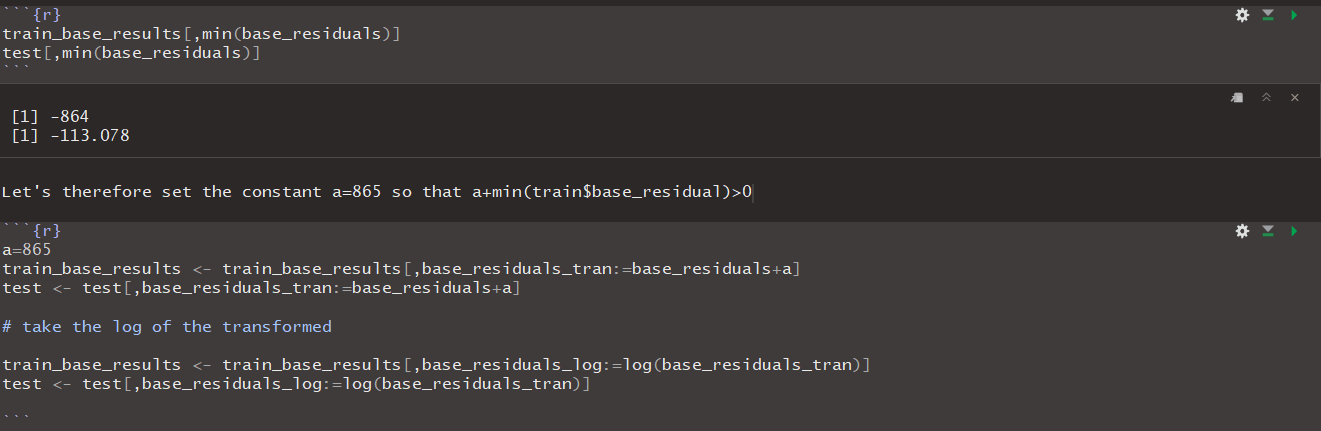
Packages Used in R

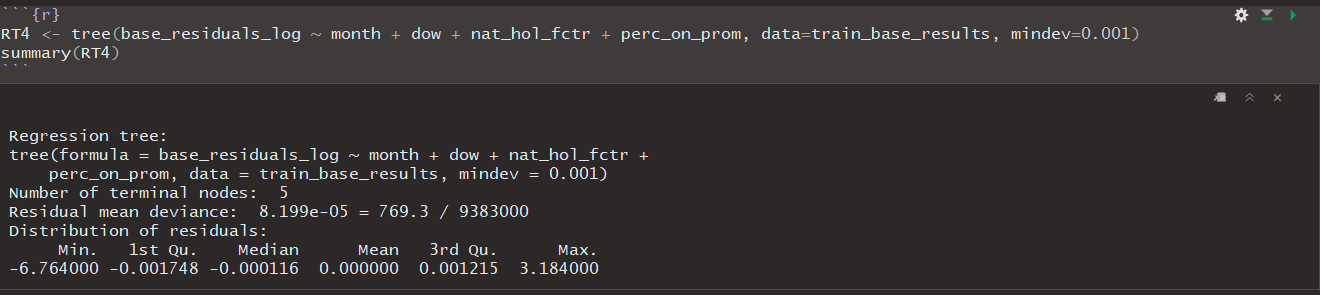
* Data.table
* Ggplot2
* Dplyr
* Lubridate
* Rpart
* Smooth
* Forecast
* Tree
* GGally

## Appendix B – Base Model R Code

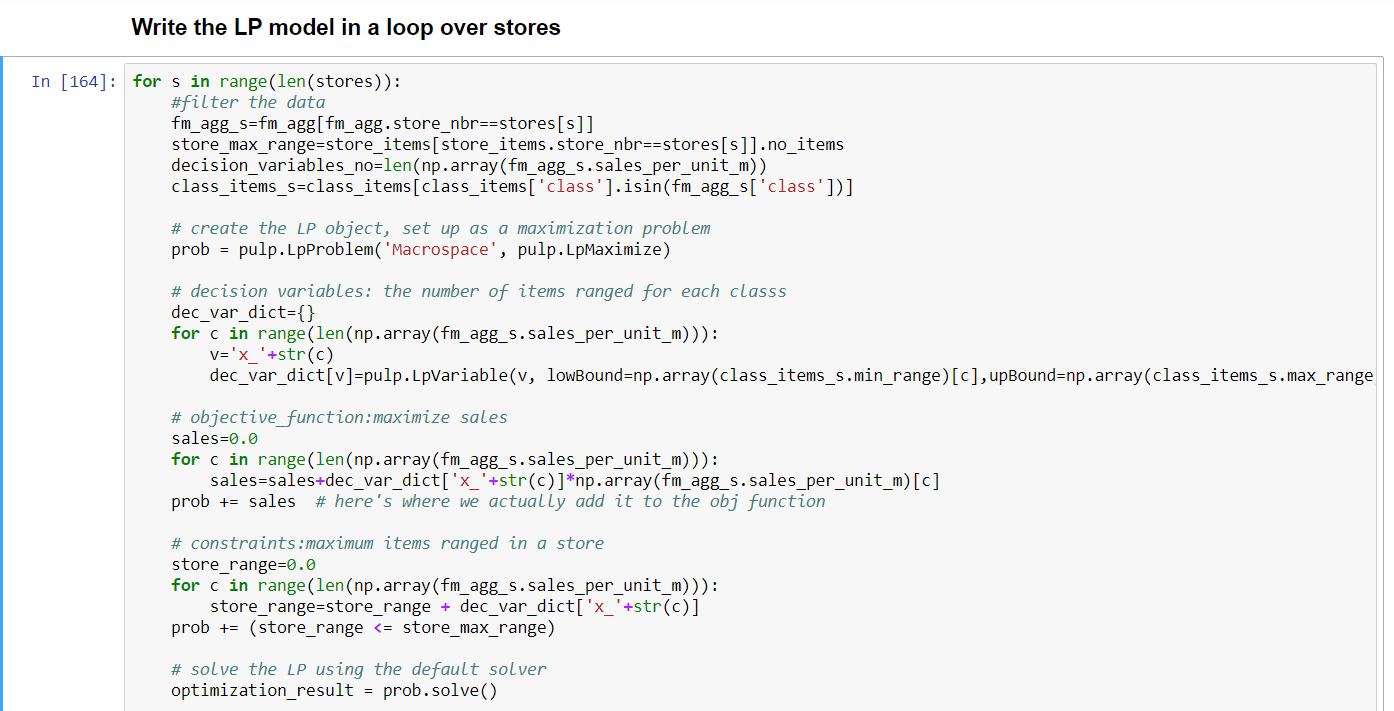


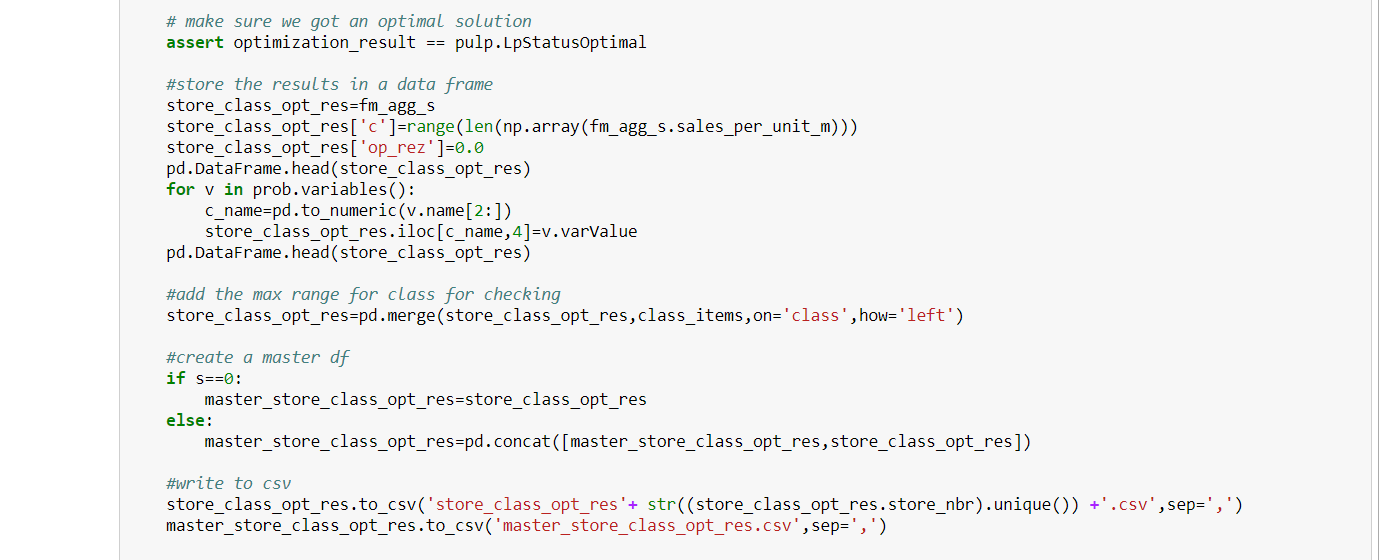
## Appendix C – Regression Tree Forecasting R Code (Final Model)





## Appendix D – Optimization Python Code





## Appendix E – Power BI Data Model

