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 are 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 how many unique products were ranged in each section-store combination as a proxy for the size of the section. This important aspect will be further discussed in section [ ].

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

In order 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 chain.

# Analysis Software

For the purpose of this analysis, I have used three main software tools. Python via jupyter notebooks for reading the data from CSV files and making transformation on the files. The reason why Python was good at this stage was because it facilitates relatively easy manipulation of very large data sets (Stack Exchange, 2015) as it does not store the information in RAM like R does. Once the data was transformed and its size reduced from 125 million rows to 12 million rows, I used R for the predictive model building and data visualization, as I was more familiar with the packages and the visualisation capabilities or R are known to be better than Python’s (Stack Exchange, 2015). For the optimization model, I have chosen Python as it’s optimization algorithm from the package ‘PULP’ was easier to understand and implement than R’s optimization package.

I have decided to use PowerBI as a visualization tool of the final results. Some of the barriers identified so far to adoption of predictive analytics by business users is the abilitity 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 via web browser. Also the visualisations are intuitive for users, with limited need for training, an example of which will be shown in Section[ ].

# Data Manipulation

## 3.1 Loading the data

The data has been loaded using python’s ‘pandas’ package, so that the output is shown as a dataframe [Figure 1].

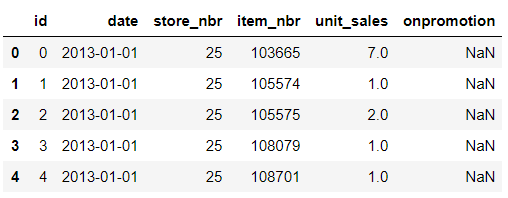


Figure 1 The first 5 rows from the ‘train’ dataset as downloaded from (Kaggle, 2017)

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

Since the ‘train’ dataset is very large, the first step would be to try and reduce it to day-store-class level from day-store-product level. For that I merged the ‘train’ and ‘items’ datasets together and checked the null values before aggregating. The only missing values were for the ‘onpromotion’ column. For deciding how to treat the missing values, I did some investigations to see if they are missing at random or if they are systematically missing (Newman, 2014). I have found that the missing values happened between January 2013 and March 2014 (Figure 2).

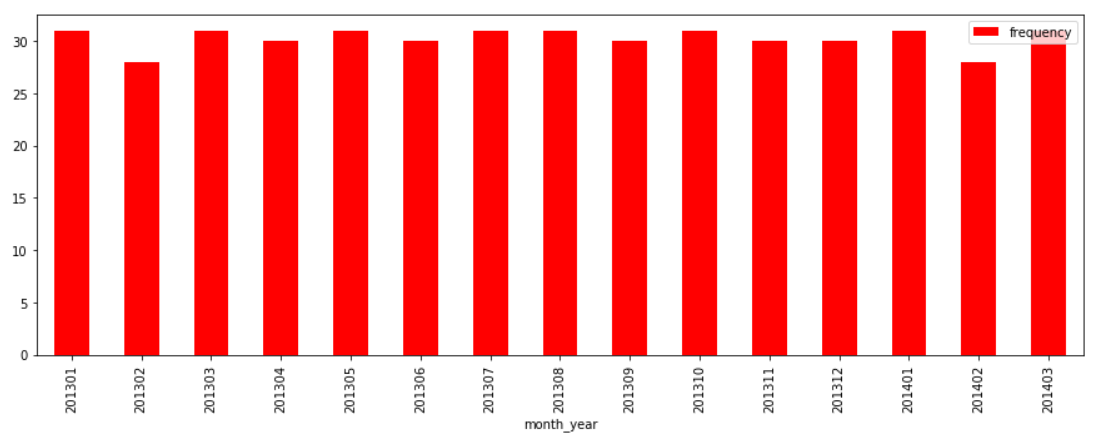


Figure 2 Number of Days with Missing Values

Since the ‘onpromotion’ feature is likely to be one of our features, it would be difficult to impute for such an extended period of time. Moreover, if the imputations are wrong, it might negatively distort the prediction algorithm. Therefore I have decided to remove these observations from my analysis and only leave data from April 2014 to August 2017, which is still a significant amount of data (Figure 4).

After merging the ‘train’ dataset with the ‘items’ dataset, I have achieved the following result:

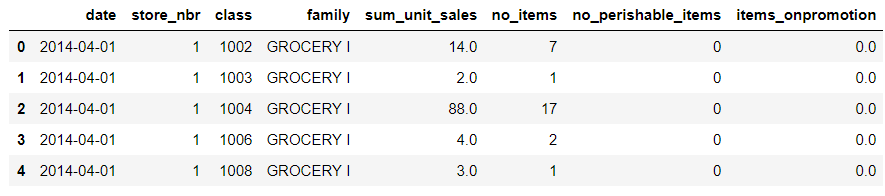


Figure 3 Train Dataset at Day-Store-Class level

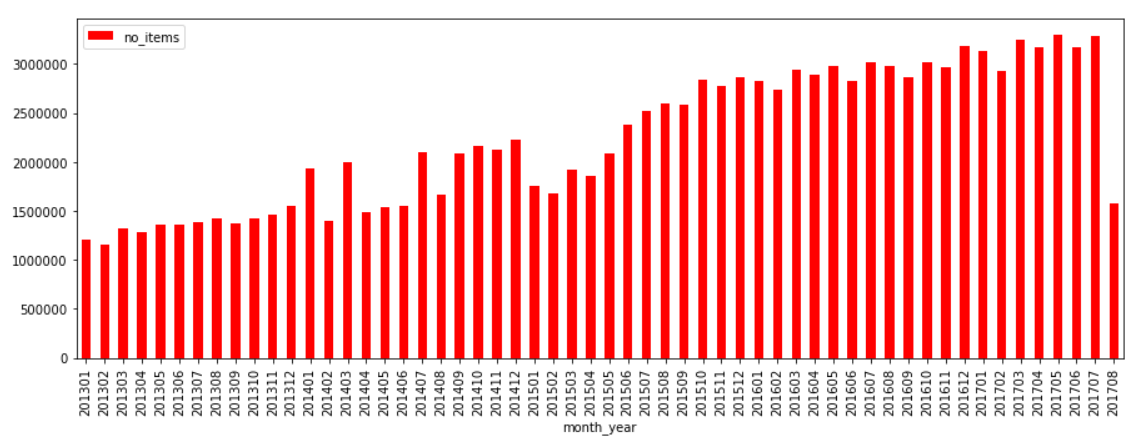


Figure 4 Number of Items ranged per Month

After the reduction, the new dataset had 12,794,614 rows, which became easier to work with in terms of adding extra features. Next, I merged this new dataset with the oil price, stores, transactions and holidays datasets to try and build derived 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, in order to create a sales per unit performance metric which is normalised so that classes with different numbers of products ranged are comparable.

Sales for a particular class is likely to depend on the type of stores. The type of stores can vary depending on the demographic profile of that store, its size, and its location in a city. In our case, although the cluster for each store and type for each store are given, their meaning is not explained on the Kaggle website.I have therefore decided to create other meaningful features for describing stores:

* how many products are ranged in each store each year (Figure 5)
* how many transactions they receive each year (Figure 6)
* what % of transactions occur during the weekdays (versus what % of transactions occur during the weekend) - this can indicate whether they are being used as top-up shopping during the week or whether they are being used as weekly shopping during the weekend. (Figure 7)
* their city coordinates - there might be shopping variations depending on each region and this will help approximate how close stores are between each other (the coordinates were taken from (Google, 2017)(Figure 8).

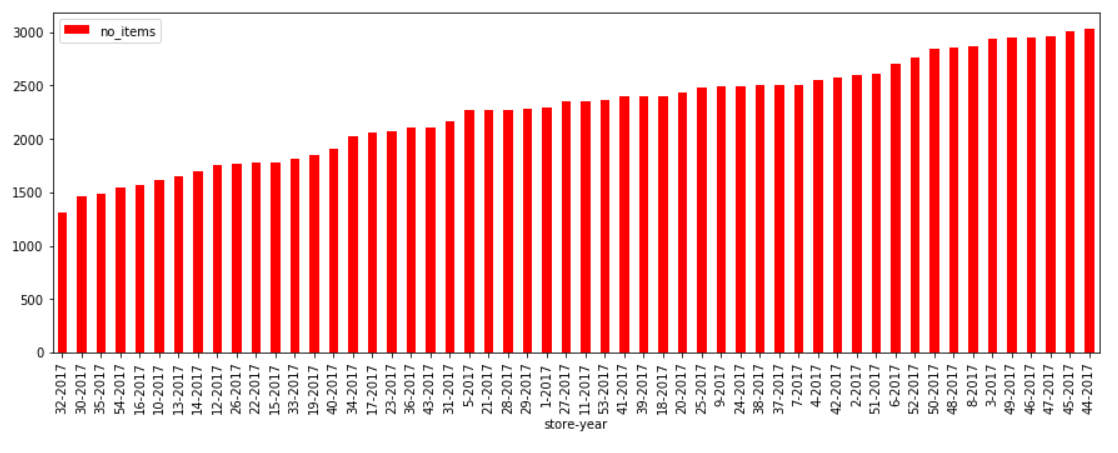


Figure 5 Number of Products Ranged Per Store-Year

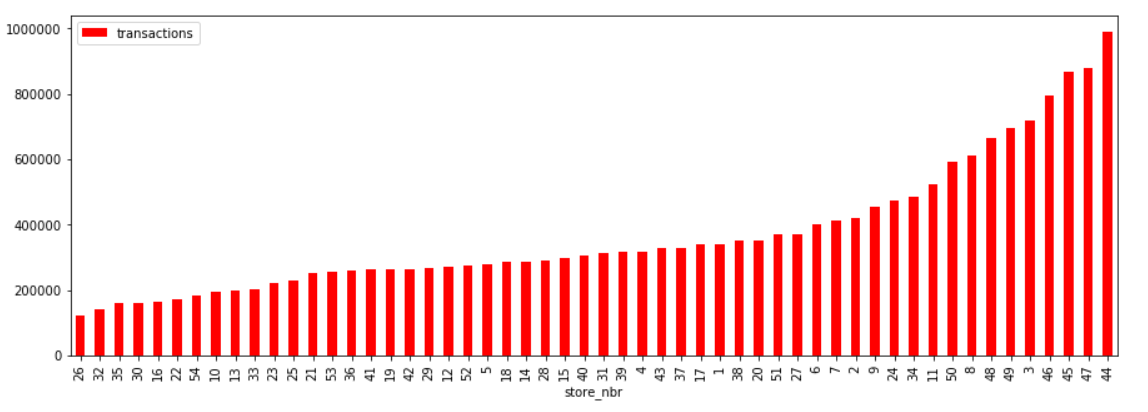


Figure 6 Number of Transactions for 2017 by Store

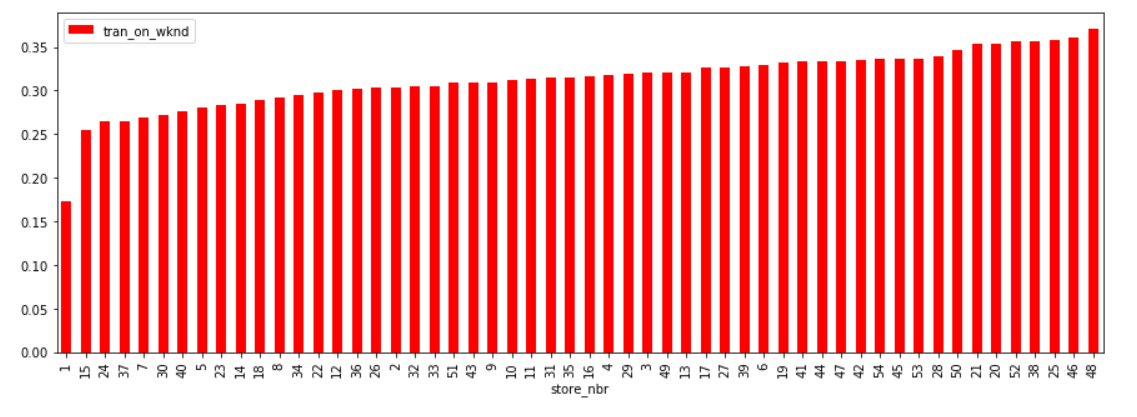


Figure 7 Percentage of Transactions happening on a Weekend

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

The wage days for the public sector in Ecuador are the 15the and last day of the month (Kaggle, 2017). I have created a wage factor feature which is 1 on wage days and decreases to 0 in equal intervals for the following 7 days after a wage. This feature recognizes the fact that consumers might increase their spending a few days after their wage days.

Similar to wage factor, I have also built national (Figure 9), regional and local holidays factor as they might be periods where consumers increase their spending in certain categories. The difference compared to wage factor is that the holidays factors build up 7 days before a holiday, gradually from 0 to 1 on the actual holiday.

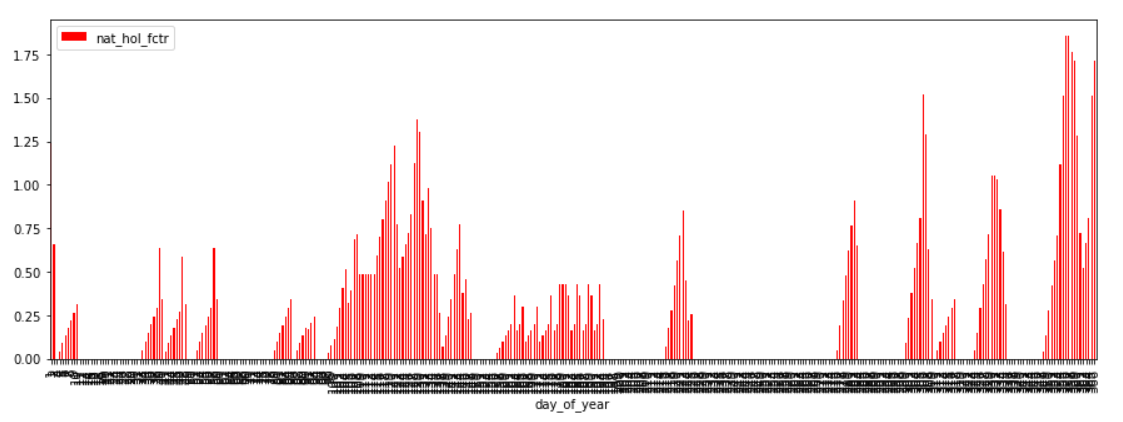


Figure 9 National Holiday factor averages for each day of year

These factors were calculated at different levels. The national holiday factor was applied to all stores, but the regional and local were only applied to those stores that were in a region or city where those holidays applied.

Next, I transformed the oil price to a monthly average. The daily oil price is not likely to have an immediate impact on the sales as it first has to impact the economy before it can translate into less spending income for consumers. Figure 10 Shows that is it difficult to find any meaningful patterns between the two variables. Moreover, the correlation between then was -0.41, which is counter intuitive as they should be positively correlated. Next, I compared the two variables at a month level, but lagging the oil price by one month as that is what we would have if we tried to predict sales a month in the future.

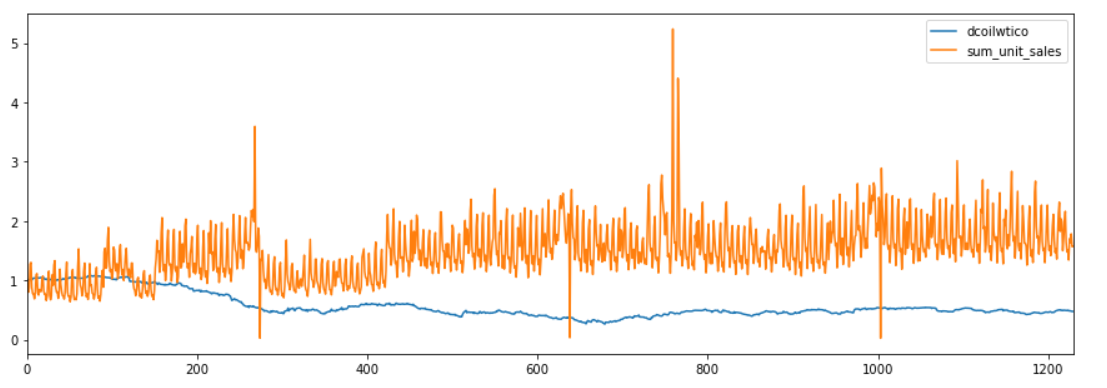


Figure 10 Daily oil price and sales fluctuations based on April 2014 as index.

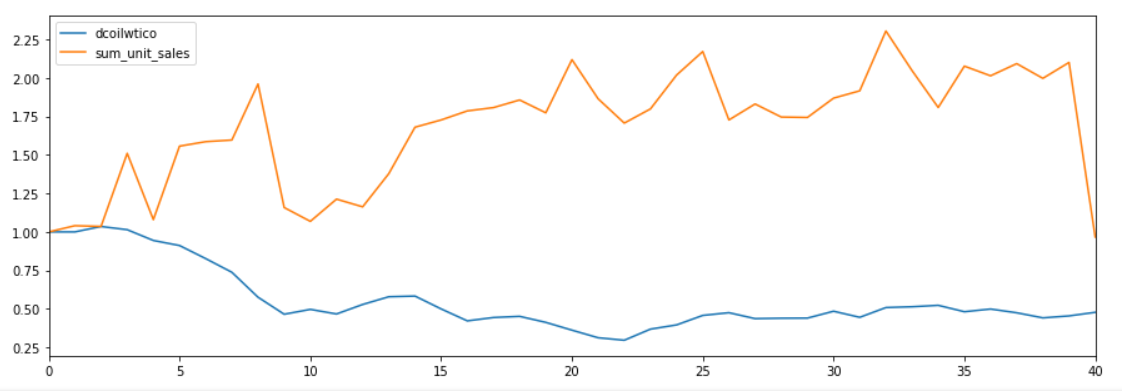


Figure 11 Monthly 1-month lag average oil price and sales fluctuations

From Figure 11 we can see that there are still no meaningful patterns and the correlation is now -0.56. I tried to lag it by two, three, or four months, but the relationship didn’t become more meaningful. Therefore, I decided to leave as a feature the one-month lagged average oil price, as it may have a stronger correlation at local levels of the dataset.

On the Kaggle website, it was also mentioned that there was a significant earthquake that happened on the 16th of April, which may have had impacts on the sales.I therefore build an earthquake factor, similar to the wage and holiday factors, that had a value of 1 on the 16th April 2016 which gradually decreased to 0 on 16th May 2016.

The other categorical features I built were the year, month, day of week and day of month to better capture time patterns. I have also derived a percentage of items on promotions for each class. This is quite difficult to predict for the following month and therefore I didn’t use it until the very few iterations of the model as I did not want to create any strong dependencies on an independent variable that might be difficult to predict one month ahead.

# Predictive Model Building

The first step when loading all the data in R was to check the data types, making sure that variables such as class or month are factors/characters rather than integer so that the predictive models interpret them correctly.

The second step was to 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 only happen in June 2017, when the latest month with full data will be May 2017.

I decided to create a computationally simple base model on the train set with which I can compare the more computationally complex models and see if the extra computational cost is worth it. The base model is an average of our target variable, sales per unit, at a year-class-store level. Therefore, the predictions for the daily store-class sales per unit for July 2017 would be the yearly averages for those store-classes combination in 2017. Before evaluating the prediction, I eliminated the store-classes combination for which the predictions are null. These are new introductions of classes in certain stores, whose performance is outside the scope of this analysis.

I have decided to evaluate models on the test set 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 more computationally complex model I tried was a simple moving average because that is often used, particularly in trading, for time series forecasting. I tried to create a forecast for each store-class combination (17,172 combinations), all with a varying degree of available data-points. 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 I chose was regression tree, as they can be used on both numerical and categorical variables and they don’t make as many assumptions as linear regression models do. I chose a simple tree rather than Random Forest as the Random Forest were 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 have also applied a log transformation on the target variable to even out the variance when predicting (Figure 12). There are still outliers, but I did not remove them as they are important for our analysis, particularly for the really high performing classes.

I then tried models with different variable combinations (Table 1).

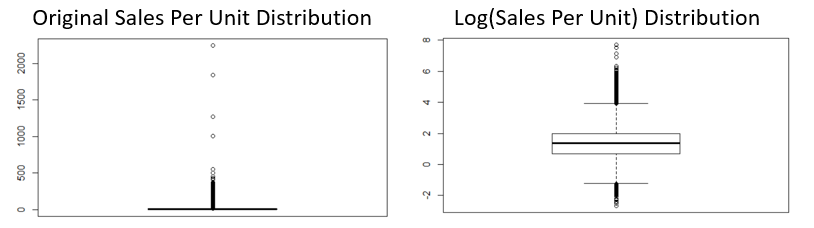


Figure 12 Set Sales Per Unit Distribution before and after log transformation.

None of the initial regression trees explained came close to the base model predictions, so I decided to build on a model on top of the base model by using a regression tree on the residuals from the base model. I first had to apply the base model on the train dataset as well. I had to remove 21% of the data-points since the base model did not have predictions of them because they were new store-class combinations in a particular month.

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 again the log transformation of residuals, but since the residuals were negative, I applied a constant to them to become 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 actor and percentage of items on promotion as the input variable (Figure 13 ).

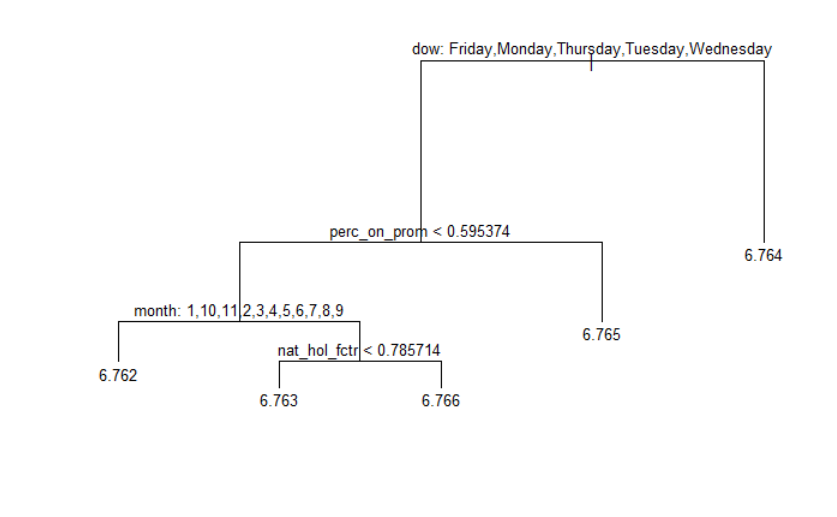


Figure 13 Final Predictive Regression Tree Model

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 prediction as an input to the optimization phase.

# Optimization

Since our decision is at a month-level, I first summed the sales per unit predictions for each store-class to get the monthly sales per unit for July 2017. Then I decided to use a linear optimization algorithm in Python from the ‘pulp’ package as I was able to define 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 sum of sales per unit. The constraints were that the number of products ranged for each class-section had to be non-negative and had to be less or equal than the maximum of unique products within a class. Additionally, the sum of total unique products in a class could not be higher than the store’s capacity, which I derived by finding the maximum number of unique products ranged for each store each year. A section of the optimization problem for a certain store is displayed in Figure 14. The optimization algorithm ran in a loop for each store, using code concepts introduced by Slavitt(2014) and Python Hosted (n.d.).

# Final Results

To understand the final results, I have pulled the unique products ranged in May 2017 for each store-class to compare to the suggested numbers for July 2017. For visualizing the results, I have created a data model in Power BI, 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.

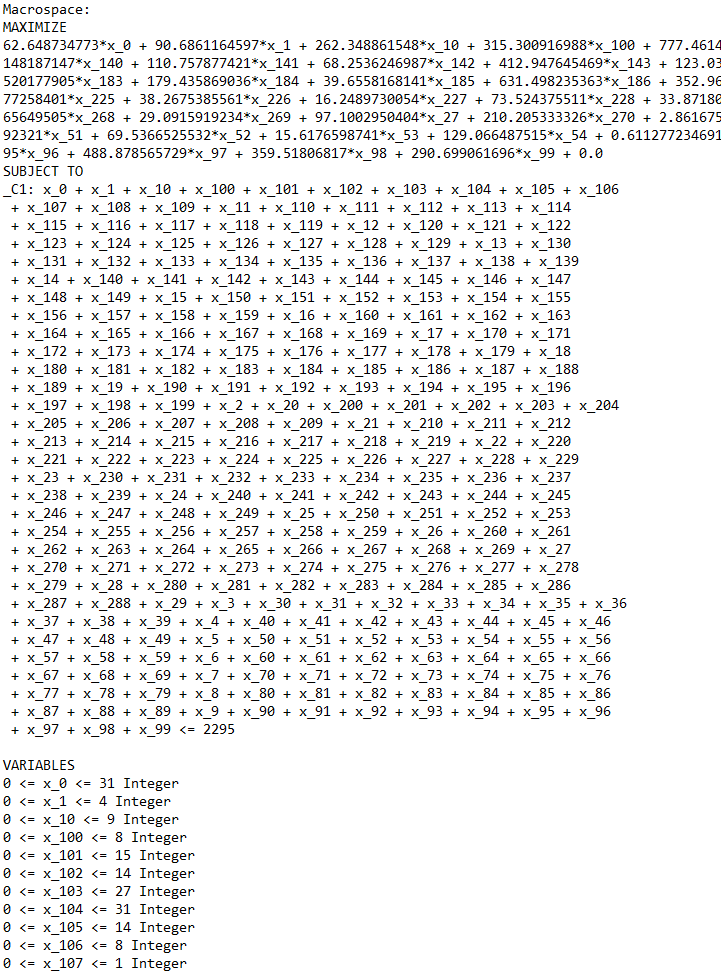


Figure 14 Optimisation Problem Description in Python's Pulp Package

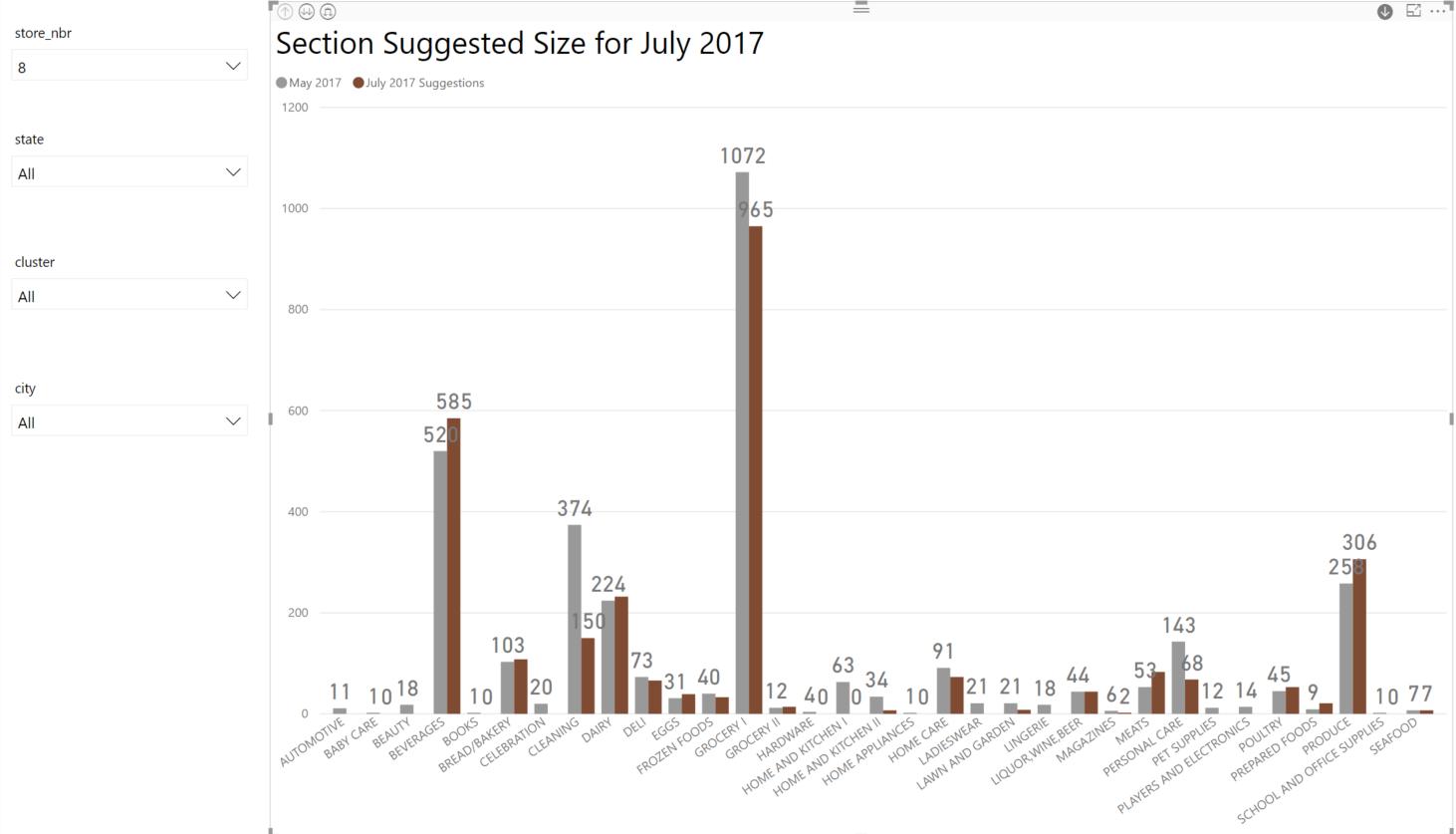


Figure 15 Power BI Visualisation of Results for Store 8 at Family/Section level

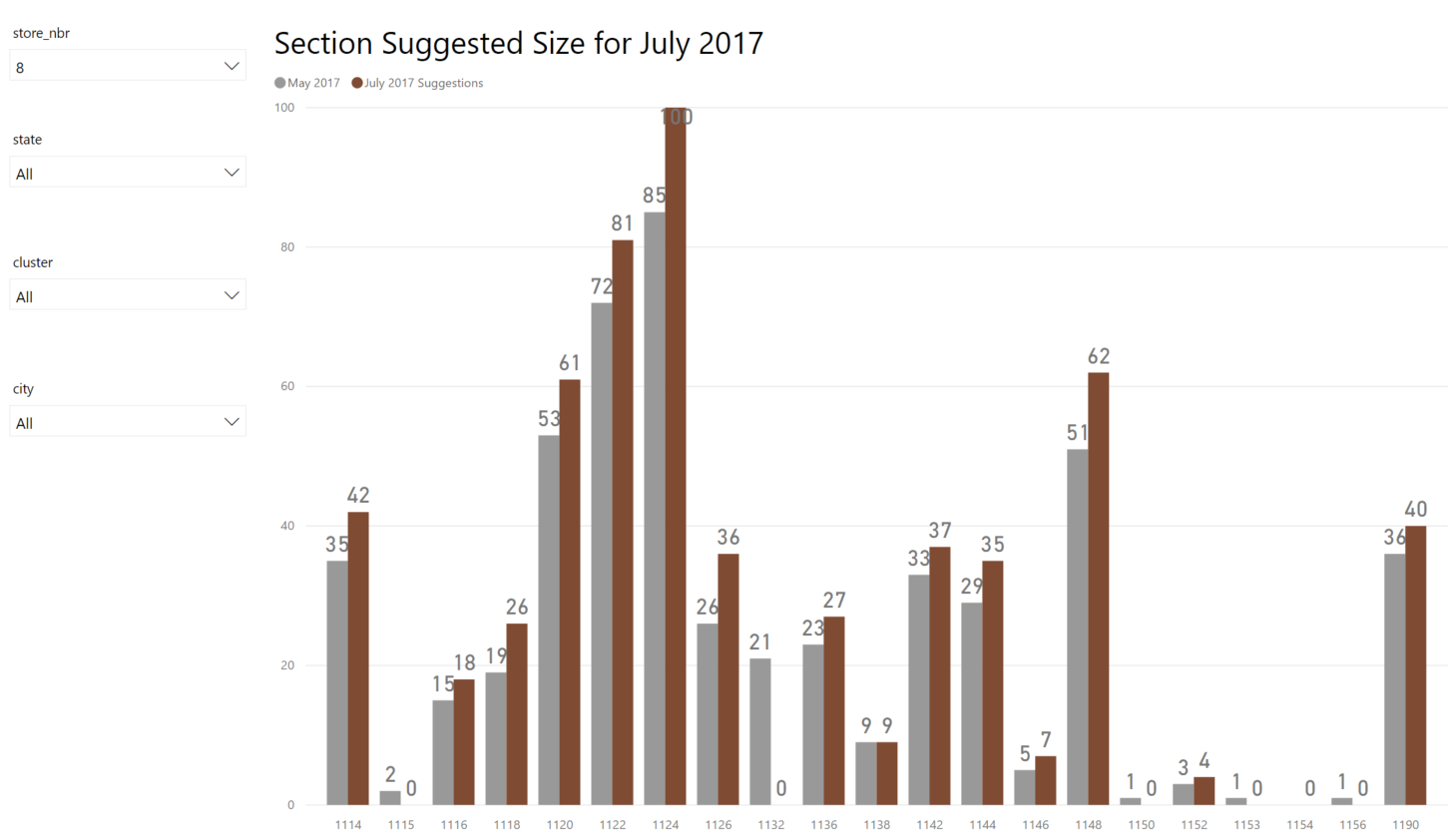


Figure 16 Power BI Visualisation of results at Class level for the Beverages section and Store 8

# Reflections

The main objective of offering decision-makers a tool that would give them 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 has to 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 have to 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 right time resources and computational resources, a better prediction model would ensure better optimization results and therefore it would give more 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 better 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 stores 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 final 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 take action 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

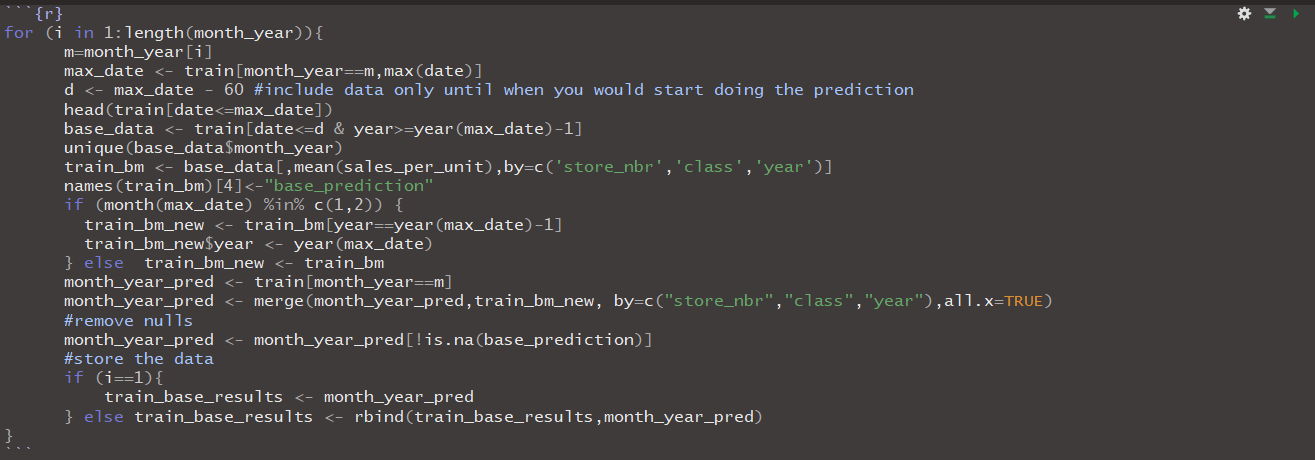
Packages used in Python:

* Pandas
* Numpy
* Matplotlib
* Pulp

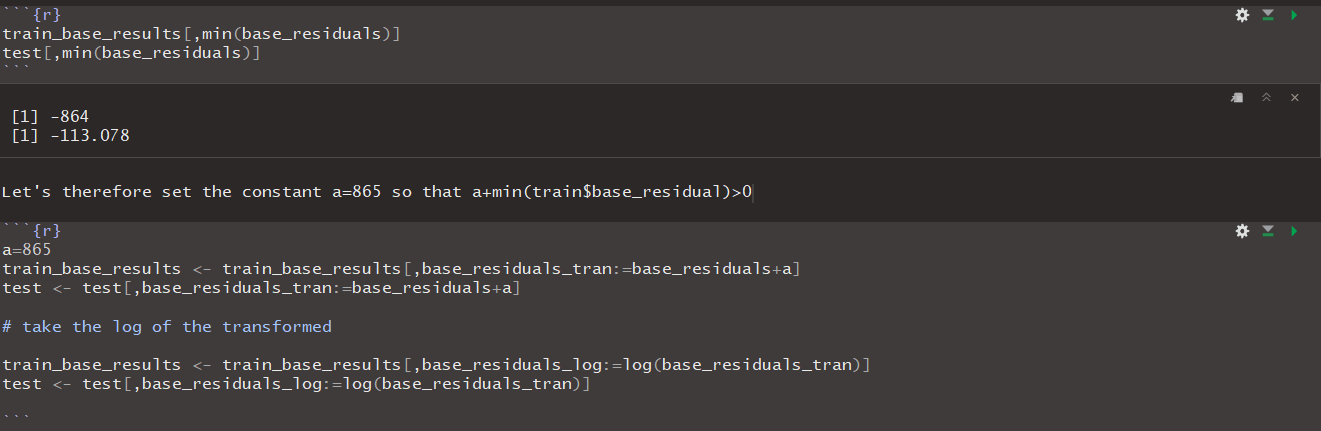
Packages Used in R

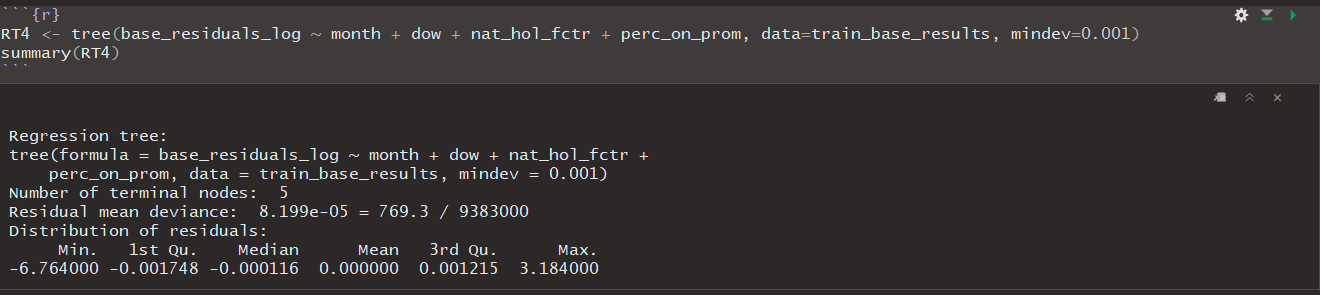
* Data.table
* Ggplot2
* Dplyr
* Lubridate
* Rpart
* Smooth
* Forecast
* Tree
* GGally

## Appendix B – Base Model R Code

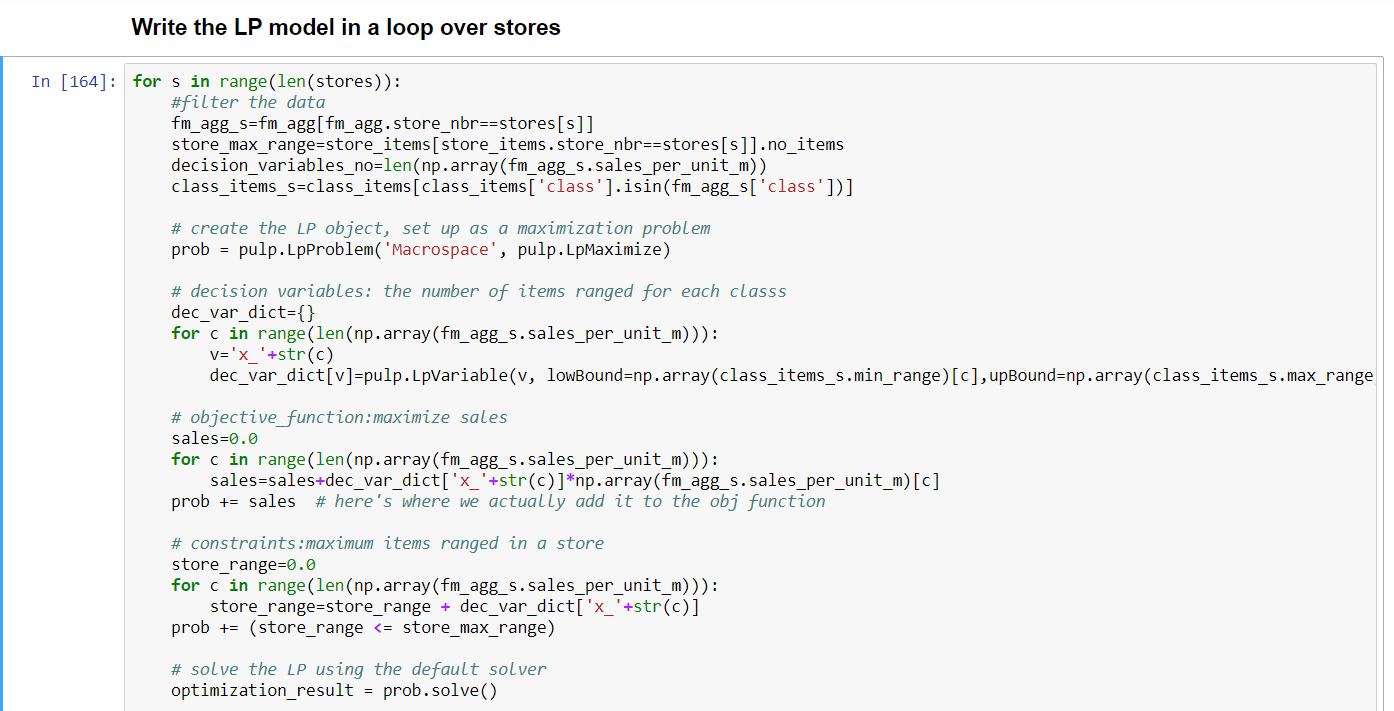


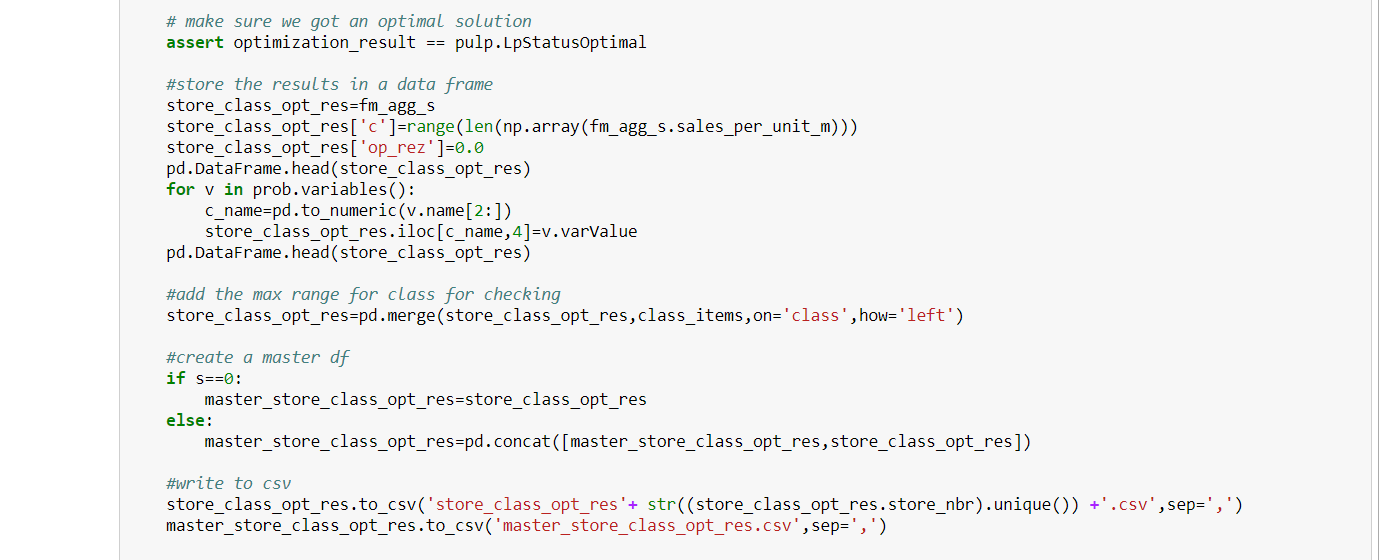
## Appendix C – Regression Tree Forecasting R Code (Final Model)





## Appendix D – Optimization Python Code





## Appendix D – Power BI Data Model

