Prediction procurement for Grocery Retailers

**Data Science Project Protocol**

Hereby submitted to Dr Tomas Karpati

By: Michael and Eyal

Date: March *13*, 2019

Table of Contents

[Introduction 5](#_Toc3224068)

[Methodology (Project design) 7](#_Toc3224069)

[Data 7](#_Toc3224070)

[The data source: 8](#_Toc3224071)

[External data sources 8](#_Toc3224072)

[Training and test 8](#_Toc3224073)

[Time frames 8](#_Toc3224074)

[Subject definition: 8](#_Toc3224075)

[Outcomes 8](#_Toc3224076)

[Confounding 9](#_Toc3224077)

[Source of bias 9](#_Toc3224078)

[Data exploration 9](#_Toc3224079)

[Enriching the data 9](#_Toc3224080)

[Outliers 10](#_Toc3224081)

[Missing values and corrupted data: 10](#_Toc3224082)

[Data retrieval protocol 10](#_Toc3224083)

[Models 11](#_Toc3224084)

[The data 11](#_Toc3224085)

[Train, validate and test 11](#_Toc3224086)

[Data balance 12](#_Toc3224087)

[Subsampling 12](#_Toc3224088)

[Technique 12](#_Toc3224089)

[Cross validation 15](#_Toc3224090)

[Measures to evaluate the model 15](#_Toc3224091)

[Assembling 15](#_Toc3224092)

[Deployment of your model 15](#_Toc3224093)

[QA 15](#_Toc3224094)

[Final user 16](#_Toc3224095)

[Presentation: 16](#_Toc3224096)

[Training and interpretation 16](#_Toc3224097)

[Platform 17](#_Toc3224098)

[Model updates 17](#_Toc3224099)

[Incomplete data 17](#_Toc3224100)

[Production models 17](#_Toc3224101)

[Results 18](#_Toc3224102)

[Amount of data 18](#_Toc3224103)

[Data balance 19](#_Toc3224104)

[Feature selection 19](#_Toc3224105)

[Outliers 21](#_Toc3224106)

[Missing values 22](#_Toc3224107)

[Prediction Results: 22](#_Toc3224108)

[Conclusion 22](#_Toc3224109)

[Annexes 25](#_Toc3224110)

[Annex 1 – GitHub 25](#_Toc3224111)

[Annex 2 – Code description 26](#_Toc3224112)

[Step 1 – Data retrieval 27](#_Toc3224113)

[Step 2 – Pair plot 27](#_Toc3224114)

[Step 3 – Box and join plots 28](#_Toc3224115)

[Step 4 – Client ID, Size 29](#_Toc3224116)

[Step 5 - Mechkar lib 29](#_Toc3224117)

[Step 6 – Z score and Outliers (outliers detection) 30](#_Toc3224118)

[Step 7 – Single product test 30](#_Toc3224119)

[Step 8 – P Value and correlation (variable selection) 31](#_Toc3224120)

[Step 9- Train& Test Balance (train test preparation) 32](#_Toc3224121)

[Step 10 – Initial prediction calculation (model selection) 32](#_Toc3224122)

[Step 10 – High correlation dataset (Variable selection) 33](#_Toc3224123)

[Step 11 – Cross validation (Fine Tune) 33](#_Toc3224124)

[Annex 3 – High level QA Protocol 34](#_Toc3224125)

[Annex 4 – Data models, results comparison 35](#_Toc3224126)

[Annex 5 – Outliers 36](#_Toc3224127)

[Annex 6 – Feature selection 37](#_Toc3224128)

[Annex 7 - Cross Validation 38](#_Toc3224129)

# Introduction

*“AI technologies could eliminate many levels of manual activities in areas such as promotions, assortments, and supply chain. AI will enable retailers to increase both the number of customers and the average amount they spend by creating personal and convenient shopping experiences.” — McKinsey Global Institute (2017)*

Seeing the number of big retail chains that are closing stores (think Sears and Payless) may support a surface-level impression that retail is dying out. But that could not be further from the truth. The industry is booming in the digital space. The revolution started by companies like Amazon and eBay has led to huge challenges for the traditional retail business model, but also massive potential for retailers and consumers alike.

This also means that there is an increasing shift towards optimisation and efficiency and a shift away from excess and waste. Retail is well-placed to benefit from the intersection of Artificial Intelligence, machine learning and big data. There is a need to manage and track a large number of items across various categories, track consumers’ shopping habits and above all, maintain a compelling brand that keeps consumers coming back. Today’s consumer wants to keep up with the latest trends, but also craves convenience; hence, the popularity of subscription boxes and online shopping. A recent survey of retailers worldwide identified cost savings, enhanced decision-making and process automation as some of the main areas that AI has the potential to impact meaningfully[[1]](#footnote-1). (Chandran, Jul 2018)

In the presented project we have done, we looked into the sales out data of a grocery retail chain that supports more the 100 franchises.

The franchises placing orders using the retailer web portal, later those orders are being sent to the manufactures or the retailer warehouse.

As can expect, the demand is varying from franchise to franchise and from time to time. A lot of causes affecting the demand, among others are price and discounts — the time of the year, holidays, weather, product availability and more.

A note that should be taking in to account is the delivery days, in the project below we measure all the parameters on a week based, since we are not dealing with the sale out of the franchises but with the sale out of the retailer to the franchises, the days of deliverables are, in most cases, fixed, i.e. manufacture X deliver its goods to franchise Y every Monday, their warehouse delivers to franchises Z every Tuesday and so on.

In this project we are focusing on the purchasing orders form a few perspectives:

We wish to identify patterns in the PO

We wish to exam how "external causes" are affecting the franchises purchasing.

We wish to build a forecasting model allowing the retailer (and its manufactures) to plan more accurately the availability of the goods

# Methodology (Project design)

## Data

The data we are using has been generated by the ERP system of the retailer and includes the sale out to its franchises in 6 months.

The data includes the franchise ID, date of the order, products ID (SKU), quantities and price.

Please note that the retailer owns the data and granted to be used in this exercise only.

In addition to it, we have used public based information such as:

* Weather
* Weeks clustering (split the data to weeks)
* Israel Holidays date

The overall the data includes more than 600,000 order lines[[2]](#footnote-2) covering a period beginning in March 2018 until August 2018.

As we refer to grocery retail, this period includes Peshach[[3]](#footnote-3) and the summer holidays[[4]](#footnote-4), two periods that require special attention.

The data included three tables (DB):

1. PO main database
2. Weather
3. Holidays

The DB's ware joined into a flat file based on the date of order[[5]](#footnote-5).

Below you will see that we have used two similar flat files that differ in the number of products each includes.

### The data source:

The main data source is the ERP system if the retailer, more data added to it from the public domain

### External data sources

We have added weather data and Holidays date.

### Training and test

Due to the size of the DB, we have used 90% of the data for turning and 10% as a test

### Time frames

The project data is of 6 months wherein the training data (90%), and the testing data (10%) are randomly chosen

### Subject definition:

In our view, the flat file data resides under inclusion criteria

### Outcomes

The main purpose of the project was to predict the demand.

Since the manufacturer's wishes (as much as possible) to be aligned with the demand, our outcomes assist to (1) define the quantities that should be manufactured to meet the demand (2) allow more accurate pre-orders by the retailer planners (in case that the products are delivered from its warehouse).

### Confounding

There are a few external confounders that can affect the results and reside in data that we don’t have. One example would be the franchise line of credit (LoC); a line of credit limits the franchises purchases that the retailer granted each one of them (open credit). The franchises are not allowed to exceed this LoC, in most cases, the LoC is managed properly; however, it might affect the purchases in special periods such as *Pesach* where the pre-holiday purchases are increasing dramatically.

### Source of bias

we don’t see any source of bias in the data we have. (no one in the supply chain wishes to have high stock volumes in its possession)

### Data exploration

As mentioned above Initially we got Excel files with the data, for data exploration and data unification we used SQL and for data visualisation we used Tableau.

The data cleansing that has been done on the SQL, then we exported a flat file to Python and continued the exploration in the notebook we used. In later stages, we have used p-value Pearson correlation and Spearman to examine the relationship and the influence of the parameters on each other (note: we are familiar with the suitability of each to the current project, even so, we found it interesting to tests them all)

### Enriching the data

We used data enrichment allowing the different models to use "more" data, the following methods have been applied on each row in the DB (see "internal" in the *Data retrieval protocol*)

1. We have defined a franchise ID based on its quantity portion out of the franchise overall quantities.
2. The quantity in week+1 (Y value)
3. Franchise size (Small or Big)
4. Weather of this week (Hi temp, Low temp and Rain Y/N)
5. Holidays, if a holiday happen at the week of the purchase (Y/N)

### Outliers

Generally, since we are dealing with real data outliers should be investigated carefully, outliers were handled based on the cause of each, errors in the data, miscalculation or any other cause.

We have identified a few outliers that although based on real data, are affecting the outcomes (minor effect) and we manage to remove them from the dataset. (see Annex 5)

### Missing values and corrupted data:

We identified approximately 170 lines (out of 600K) with corrupted data or missing values, those lines are mostly in the boundaries of the data, and therefore we decide to remove them from the dataset

### Data retrieval protocol

(see at GitHub <https://github.com/eyalfarkash/TCDS-Final.git> under Documents, Data Retrieval Protocol V4.xlsx)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Feature name** | **Source** | **Value type** | **Scale** | **Data type** | **Unique count** | **Measure unit** | **Conversion factor** | **Min** | **Max** | **Outlier treatment** | **Null** | **Notes** |  |  |  |  |
| Product | PO | Numeric | Nominal | Varchar | Yes | N/A | N/A | 0 | 7.3E+12 | Leave | NULL | SKU number |  |  |  |  |
| Weenum | PO | Numeric | Nominal | Integer | No | N/A | N/A | 9 | 36 | Leave | NA | week number |  |  |  |  |
| Above\_Standart\_Price | PO | Numeric | Nominal | Boolean | No | N/A | N/A | 0 | 1 | Leave | NULL |  |  |  |  |  |
| Below\_ Standart\_Price | PO | Numeric | Nominal | Boolean | No | N/A | N/A | 0 | 1 | Leave | NULL |  |  |  |  |  |
| Sum\_Quantity | PO | Numeric | Nominal | Integer | No | Unit | N/A | 0 | 10000 | Null | NULL | total quantity in a week | |  |  |  |
| Standard\_Price | Product | Numeric | Nominal | Float | No | N/A | N/A | 0 | 1000 | Leave | NULL | Standard price |  |  |  |  |
| Holiday | Holiday | Categorical | Nominal | Boolean | No | N/A | N/A | 0 | 1 | Null | NULL |  |  |  |  |  |
| Temp\_Hi | Weather | Numeric | Nominal | Float | No | N/A | N/A | 10 | 50 | Leave | NULL |  |  |  |  |  |
| Temp\_Lo | Weather | Numeric | Nominal | Float | No | N/A | N/A | 0 | 25 | Leave | NULL |  |  |  |  |  |
| Temp\_Rain | Weather | Numeric | Nominal | Boolean | No | N/A | N/A | 0 | 1 | Leave | NULL |  |  |  |  |  |
| Avg\_Price\_Change | Internal | Numeric | Nominal | Float | No | N/A | N/A | -1000 | 1000 | Leave | NULL |  |  |  |  |  |
| Sum\_Quantity\_1\_Y | Internal | Numeric | Nominal | Float | No | N/A | N/A | 0 | 100000 | Leave | NULL | The sale quantity in Week+1 (it is our Y) | | |  |  |
| Avg\_Price\_Change\_1 | Internal | Numeric | Nominal | Float | No | N/A | N/A | -1000 | 1000 | Leave | NULL | avg price change in week+1 | |  |  |  |
| All\_week | Internal | Numeric | Nominal | Integer | No | N/A | N/A | 0 | 39 | Leave | NULL | Number of weeks that the franchise is operating | | |  |  |
| Count\_week\_PO | Internal | Numeric | Nominal | Integer | No | N/A | N/A | 0 | 39 | Leave | NULL | Number of weeks that the franchise bought this product (SKU) | | | |  |
| Sum\_Quantity\_1\_minus\_Sum\_Quantity | Internal | Numeric | Nominal | Integer | No | N/A | N/A | 0 | 1000 | Leave | NULL | the total quantity of next week minus this week (the change) | | | |  |
| Avg\_Quantity\_for\_Week | Internal | Numeric | Nominal | Float | No | N/A | N/A | 0 | 1000 | Leave | NULL |  |  |  |  |  |
| Avg\_Quantity\_for\_PO | Internal | Numeric | Nominal | Float | No | N/A | N/A | 0 | 100 | Leave | NULL |  |  |  |  |  |
| Total\_Quantity\_for\_Customer | Internal | Numeric | Nominal | Integer | No | N/A | N/A | 0 | 100000 | Leave | NULL |  |  |  |  |  |
| Total\_Quantity | Internal | Numeric | Nominal | Integer | No | N/A | N/A | 0 | 100000 | Leave | NULL |  |  |  |  |  |
| Frequency\_PO | Internal | Numeric | Nominal | Integer | No | N/A | N/A | 0 | 10 | Leave | NULL | how often a PO for the product is issued | | |  |  |
| Customer\_ID | Internal | Numeric | Nominal | Float | Yes | N/A | N/A | 0 | 100 | Leave | NULL | it is a relative number; the customer ID is its quantities portion out of all quantities | | | | |
| Customer\_Number | PO | Numeric | Nominal | Integer | Yes | N/A | N/A | 416000000 | 416000170 | Null | NULL |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## Models

### The data

We have initiated two datasets and compared the results between them:

* 75-products database: the DB includes the app 19% of the orders performed by the franchises, overall approximately 110K lines
* 3-Products database: the DB is a subset of the 75 products DB, wherein only three products exist, comprise approximately 9K orders

### Train, validate and test

As mentioned above, we have analysed the data on a weekly base, we have operated in the following models on the data:

Randomly chosen: 90% for the train while the other 10% used for dev/test.

### Data balance

The data balance of the 3-product dataset has been checked based on the number of PO per product. See more on the results section.

### Subsampling

The data we use is divided into franchises; however, a sample data of each does not represent its purchasing history and trends; therefore stratified sampling is not applicable for our data subsampling

### Technique

We have used the following techniques

* Linear regression
* Lasso
* Ridge
* SVM
* Ada boost and
* Logistic regression

[Note: we are aware that some of the above methods are not aligned with the examined data; however we found it interesting to check those too and to present the differences, below you would see that at least one of the “nonaligned” method provides rational results.]

we compare between them using the full dataset as presented below:

In the data side we had, the POs with the following parameters:

* Customer\_ID – represent the customer ID, the ID is calculated based on the franchise size
* Weeknum – week number of the PO
* Product - product SKU
* Customer\_Number - the franchise number
* Sum\_Quantity – the ordered quantity of Product
* Avg\_Price\_Change – Average price of the Product
* Avg\_Price\_Change\_1 – Average price of the product in the following week
* Sum\_Quantity\_1\_minus\_Sum\_Quantity – the quantity difference between this and the next week order
* Avg\_Quantity\_for\_Week - Average quantity per week
* Avg\_Quantity\_for\_PO – the average quantity per PO
* All\_week – how many weeks the franchise was open
* Count\_week\_PO – in how many weeks the franchise placed an order
* Standard\_Price - the price of the product
* Above\_Standard\_Price - in case that in this PO the price is higher than the standard price
* Below\_Standard\_Price – in case that in this PO the price is lower than standard price (for example special offer)
* Total\_Quantity\_for\_Customer - The total quantity of this Product to this franchise
* Total\_Quantity - the total quantity of this product
* Holiday – if a holiday is at this week
* Temp\_Hi - the average high temperature (of this week)
* Temp\_Lo - the average low temperature of this week
* Temp\_Rain – was it rainy this week
* Frequency\_PO - what was the frequency of PO of this franchise
* Sum\_Quantity\_1\_Y - the quantity ordered of this product in the following week – this is the Y parameter we are looking for

Dataset: **Data** includes all the flat file parameters

* Customer\_ID float64
* Weeknum int64
* Product int64
* Customer\_Number int64
* Sum\_Quantity int64
* Avg\_Price\_Change float64
* Avg\_Price\_Change\_1 float64
* Sum\_Quantity\_1\_minus\_Sum\_Quantity int64
* Avg\_Quantity\_for\_Week float64
* Avg\_Quantity\_for\_PO float64
* All\_week int64
* Count\_week\_PO int64
* Standard\_Price float64
* Above\_Standard\_Price int64
* Below\_Standard\_Price int64
* Total\_Quantity\_for\_Customer int64
* Total\_Quantity int64
* Holiday int64
* Temp\_Hi float64
* Temp\_Lo float64
* Temp\_Rain int64
* Frequency\_PO float64
* Sum\_Quantity\_1\_Y int64

Using the two best models (Linear and Adaboost), we perform a re-scoring using two additional datasets, DropData and DaropData2 with the following parameters:

**DropData** includes the following parameters based on Spearman, Pearson and p-value.

* Customer\_ID float64
* Product int64
* Sum\_Quantity int64
* Avg\_Price\_Change float64
* Avg\_Price\_Change\_1 float64
* Sum\_Quantity\_1\_minus\_Sum\_Quantity int64
* Avg\_Quantity\_for\_Week float64
* Avg\_Quantity\_for\_PO float64
* Count\_week\_PO int64
* Standard\_Price float64
* Total\_Quantity\_for\_Customer int64
* Sum\_Quantity\_1\_Y int64

**DropData2** is a subset of DropData where we tried to find manually, additional un-necessary parameters:

* Customer\_ID float64
* Product int64
* Sum\_Quantity int64
* Avg\_Price\_Change float64
* Avg\_Price\_Change\_1 float64
* Sum\_Quantity\_1\_minus\_Sum\_Quantity int64
* Avg\_Quantity\_for\_Week float64
* Count\_week\_PO int64
* Total\_Quantity int64
* Sum\_Quantity\_1\_Y int64

See comparison and examination results on Annex-6

### Cross-validation

Since we have a limited amount of data, we have used 2,5,10,15,20,25,30,40 and 50 (K) fold cross-validation on the models (see Annex-6).

Additionally, as you would see the most accurate method found was Linear regression, and therefore we have added two methods, Ridge and Lasso.

### Measures to evaluate the model

We are using mean absolute error (MAE) to measure the difference between the actual values and the estimated values.

Since we are calculating "large numbers" (i.e. the total quantity of a product that has been bought by all the franchises together) the estimation can have an error, however, it is important to understand its magnitude and if it's positive or negative (overstock or out of stock).

We find this method as the most appropriate to use while dealing with finite numbers.

### Assembling

We have tested based on the well-known models that fit structured data and used the most accurate one.

We chose not to use assembling since we got a much higher result in one technique comparing to the others (see more in annex-4, Data model results comparison)

## Deployment of your model

### QA

The QA of the project shall be performed by a professional QA team with knowledge in the field and the data.

In our case, the procurement team is responsible for the orders. However, it is important that the commercial/marketing department will give its inputs since special offers, promotions and product availability may affect the prediction

QA protocol see in annex-3

### Final user

We have two final users (1) the manufactures and (2) the retail chain planner;

* The manufactures – they are planning the production based on the demand, better forecasting of the demand shall optimise the production and supply chain operation.
* The chain planner – the chain planned to place orders on behalf of the warehouse to the manufactures, a warehouse procurement is different from the end user, a purchasing order should be of high volume and for a long duration, an accurate planning of the warehouse reduce the overstock or out of stock of the warehouse, in addition, it assists in stock management in terms of expiration dates and such.

### Presentation:

Using the model. A planner can enter the data it has, such as product number, store ID, planned week, price act. And get the forecast of this franchise/product order quantity.

### Training and interpretation

We don't think that it would be necessary, an expert in the field would understand the outcomes.

### Platform

At this stage of the project, as we are dealing with a relatively small amount of data and for a limited number of users. A standard computer\server with the needed environment installed will do the job.

In the future, if necessary, it can be exported to cloud services utilising GPU's[[6]](#footnote-6).

### Model updates

It is difficult to predict since it related very much to the "real life" results it provides. As the amount of data increase and the model has more data to learn from, we believe that it would become more accurate. In case that it will not happen then a reevaluation of the model would be needed.

### Incomplete data

In the case of incomplete data, it is most likely that the SW will error and notify the user, for the data that we are using there are no defaults values that we can use.

Defaulted values such as average will generate noise in the data, and too much of those may generate bias and affect the prediction.

### Production models

Using the full data set, including the following parameters:

Customer\_ID float64

Weeknum int64

Product int64

Customer\_Number int64

Sum\_Quantity int64

Avg\_Price\_Change float64

Avg\_Price\_Change\_1 float64

Sum\_Quantity\_1\_minus\_Sum\_Quantity int64

Avg\_Quantity\_for\_Week float64

Avg\_Quantity\_for\_PO float64

All\_week int64

Count\_week\_PO int64

Standard\_Price float64

Above\_Standard\_Price int64

Below\_Standard\_Price int64

Total\_Quantity\_for\_Customer int64

Total\_Quantity int64

Holiday int64

Temp\_Hi float64

Temp\_Lo float64

Temp\_Rain int64

Frequency\_PO float64

Sum\_Quantity\_1\_Y int64

We ran it on the following models:

* Linear regression
* Lasso
* Ridge
* SVM
* Ada Boost
* Bayes
* Logistic regression

Based on the results (as presented in Annex-4) we identify the best algorithm as a lasso, to optimise the results, we used cross-validation and cross predict, in it, we riches a Mean Absolute Error value of 4.04.

Additional datasets that ware used are presented in Results\Features selection

# Results

## Amount of data

As I have mentioned we have used two datasets, (1) 75 products and (2) 3 products:

The 75 products dataset has 107K lines for train and 11K lines for test

The 3 Products dataset has 7353 lines for train and 817 lines for test

### Data balance

As you would see in the results, the 3-product dataset generated a better result. To make sure that those are the optimal results the train and test datasets were split randomly and then we have checked the balance of each product in the train and test. Keeping in mind that the actual number can be changed between the run’s, when we tested it, in most cases, it was well balanced (all products ware between 32%-36% of the dataset[[7]](#footnote-7)). Also, note that since the Y parameter is not a category but a number (quantity) we preferred to keep some additional data over adding manipulated data (that might affect the outcomes) or perform data reduction (that limit the train).

The data balance is in the "Train&Test Balance" section in the *final-project-CSV\_v9\_3Products* notebook

### Feature selection

Initially, we have run the model using all the available parameters as described in the data retrieval protocol, i.e. 23 parameters retrieved from 3 datasets plus internal parameters that ware calculated by us to support the model.

The initial assumption was that the model would be most accurate by using the maximum number of parameters.

To examine this assumption, we have calculated the Spearman, Pearson correlation and a p-value of all parameters and in particular with relation to the Y parameter.

The results were as follow:

**P Value**  **Pearson Spearman**

Customer\_ID 0 0.245 0.272

Weeknum 0.87 0.0018 0.032

Product 0 -0.146 -0.268

Customer\_Number 0.0123 0.0277 0.058

Sum\_Quantity 0 0.2354 0.139

Avg\_Price\_Change 0 -0.2933 -0.255

Avg\_Price\_Change\_1 0 -0.2951 -0.285

Sum\_Quantity\_1\_minus\_Sum\_Quantity 0 0.7819 0.780

Avg\_Quantity\_for\_Week 0 0.610113 0.584

Avg\_Quantity\_for\_PO 0 0.5998 0.559

All\_week 0.001 0.0364 0.048

Count\_week\_PO 0 0.2036 0.242

Standard\_Price 0 -0.2942 -0.268

Above\_Standard\_Price 1 -0.0048 -0.002

Below\_Standard\_Price 0.658 0.1246 0.164

Total\_Quantity\_for\_Customer 0 0.2454 0.272

Total\_Quantity 0 NA NA

Holiday 1 -0.030 -0.045

Temp\_Hi 0.0061 -0.006 0.037

Temp\_Lo 0.575 0.0007 0.045

Temp\_Rain 0.948 -0.039 -0.037

Frequency\_PO 0.0004 -0.186 -0.256

Sum\_Quantity\_1\_Y 0 1 1

It is seen in Spearman, P-Value and Pearson that a few parameters have a low

correlation to the Y parameter and as of that we developed two additional datasets for tests, dataset "**Drop Data**" that includes the high related parameters:

* Customer\_ID float64
* Product int64
* Sum\_Quantity int64
* Avg\_Price\_Change float64
* Avg\_Price\_Change\_1 float64
* Sum\_Quantity\_1\_minus\_Sum\_Quantity int64
* Avg\_Quantity\_for\_Week float64
* Avg\_Quantity\_for\_PO float64
* Count\_week\_PO int64
* Standard\_Price float64
* Total\_Quantity\_for\_Customer int64
* Sum\_Quantity\_1\_Y int64

And "**Drop Data 2**" wherein we removed additional parameters that we thought (wrongly)

that will support the model above the "Drop Data" parameters[[8]](#footnote-8)

* Customer\_ID float64
* Product int64
* Sum\_Quantity int64
* Avg\_Price\_Change float64
* Avg\_Price\_Change\_1 float64
* Sum\_Quantity\_1\_minus\_Sum\_Quantity int64
* Avg\_Quantity\_for\_Week float64
* Count\_week\_PO int64
* Total\_Quantity int64
* Sum\_Quantity\_1\_Y int64

As of the results, we learned that there are parameters that do not affect the result for

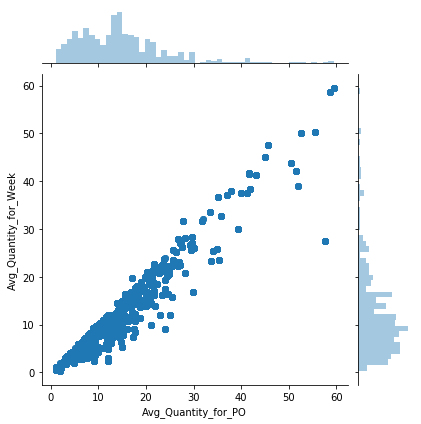
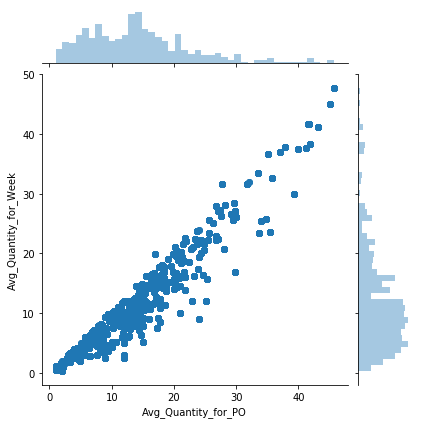
example *Rain*, while others are affected it negatively, for example: *Below standard price*.

At the end "Drop Data" provides the most accurate results.

The results of the different datasets are presented in the cross-validation annexe (Annex – 6 below)

## Outliers

We didn’t have so many outliers in the data due to its source; however, we have removed a few outliers (see annexe 5) that had a minor impact on the results as can be seen in the plots below and the notebook. The plot presents the Avg quantity per PO vs quantity per week; it can be seen that in most cases the Avg PO is lower than 50 while the higher values are "out of the function" and therefore can be treated as outliers. Additional outlier mainly caused by the Pesach holiday, the sales of a product increased dramatically in the week before the holiday and caused the sum\_quantity parameter to increase to "outlier levels", (in the data we saw a gap in the quantities) the action we took was to remove the sales that exceed 190 units, making the sum\_quantity more homogeneous over the whole period. To support this method of operation we have calculated the average, mean, standard deviation and z-score of those parameters. We didn’t limit ourselves to 2 or 3 standard deviations since such action might be too significant (after the peak comes the fall and is not negative i.e. it doesn’t balance the peak), keeping in mind that the data we have is real and reflects the activity, removing those peaks may cause bias.

Plot 1: Outliers before and after

## Missing values

In the whole dataset, we had app 170 lines with missing values since the values ware mostly prices and where not a part of the 3 Product dataset we deleted those lines from the initial dataset.

## Prediction Results:

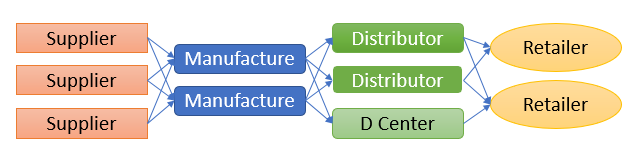
In the 75-product dataset, we managed to get a mean absolute error of 14

While in the 3 Products we managed to get a mean absolute error of 4.04

# Conclusion

In this project, we have developed a proof of concept for *prediction procurement*.

The retail market and in particular the grocery market relays on product availability (the right product at the right time in the store), distributors deliver those products and manufacturers. The distributors rely on the manufacturers, suppliers and so on, causing a long supply chain between the supplier and the retailer with lead times in between.



The immediate result of the above is: It is important to plan and to place the current order at the right time.

When we initiated the project, after gathering the data, we understood that the data "as-is" is insufficient and we need to take care of it, as *"the quality of the results = the quality of the data"*.

From the retailer we got the data in excel sheets, trying to "play" with it in MS-Excel was intricate. We understood that a data environment is needed for data cleansing and therefore we transferred it to SQL DB that allowed us to manipulate it more efficiently while using Tableau as a data visualisation tool.

Also, we saw the need (with your assistance) to add more variables to the dataset, external variables such as weather and Holidays have been added and internal such as average-price, average quantity and more. In the end, a unified flat file for the use of the model was generated

The next phase was to build the python SW that supports the model, at this stage, again, we looked into the data, exploring animalities or mistakes that might affect the results or the operation itself.

Upon finalisation of the data, we have ren a few supervised models and compare the results, we looked into the accuracy of the results and at the error rate. Since we are dealing with grocery, a non-accurate value with a low error can be sufficient[[9]](#footnote-9)

We saw that the outcomes of accuracies are relatively low. While reviewing the dataset, we found that there is a low correlation between the different products and that each acting differently, therefore we have decided to check if a smaller dataset can result in more accurate outcomes. We generated a 3-products dataset for tests (as of that point we used a 75 products dataset).

As expected, the 3-products dataset, although it is a smaller one (less than 10% of the original) it generates more accurate results on both accuracy and error-rate.

Then we have initiated two operations: (1) remove low correlation parameters while testing using the different models and a (2) cross-validation operation. While the first didn’t prove itself dramatically in the second, we got a lower error rate

It is worth noting that the most accurate model we used was Lasso, that is used as a general approach for many statistical models.

Of course, like any other machine learning algorithm it has its limitations, the main limitation is the data, outliers and noise, in the model that we have developed we were highly concern about the data quality and spent a large effort on its cleansing, in a production environment it is not always the case, in addition: (1) it might be that different products or at a different stage in the supply chain act differently and (2) the number of products, as it seems to us, the model is more suitable for low number of products and therefore we are not recommending using it with a large number of products.

**To conclude**: In this project, we manage to show the advantages of *procurement prediction* based on machine learning, the results that we got are a good starting point in the journey towards highly accurate errors free prediction model. We have presented the limitations we faced along the way and the methods we used. At this stage, it cannot be used as a "single source of trues" product to be used by retailers, but an assistance tool or a step in the direction towards the *automatic orders* system

# Annexes

## Annexe 1 – GitHub

All the files that have been using for this project reside in Github and can be view by using the following link: <https://github.com/eyalfarkash/TCDS-Final.git>

The git includes the following directories:

* **Code**: in it, you will find the code we develop for the project:
  + Final-project-CSV\_v4\_FullData-final.ipynb – was used on top of the 75 products dataset
  + Final-project-CSV-v9\_3Products-final.ipynb – was used on top of the three products dataset
  + SQLQueryFin.sql – includes the SQL code used for the generation of the flat files
  + All the other files are drafts and tests we have done.
* **Data**: includes the data files we use
  + 3Products.CSV is the 3products dataset
  + Final\_csv1\_full\_v3 is the 75 products dataset
  + Hebrew holidays used to identify the Israeli holidays' dates
  + Weather2018 used for day and night temperatures and rain
* **Documents**: includes some documentation on the project
  + The EDA directories are of the Mechkar HTML pages
    - [3Product-EDA-Mechkar-V4](https://github.com/eyalfarkash/TCDS-Final/tree/master/Documents/3Product-EDA-Mechkar-V4) is of the 3 Products
    - [Full-EDA-Mechkar-HTML-V3](https://github.com/eyalfarkash/TCDS-Final/tree/master/Documents/Full-EDA-Mechkar-HTML-V3) is of the 75 Products
  + [Data Retrieval Protocol V4.xlsx](https://github.com/eyalfarkash/TCDS-Final/blob/master/Documents/Data%20Retrieval%20Protocol%20V4.xlsx) is the retrieval protocol
  + TCDS Project Protocol V6 – is this document
* **Ref-Code**: is a draft directory we place in the git for our internal use

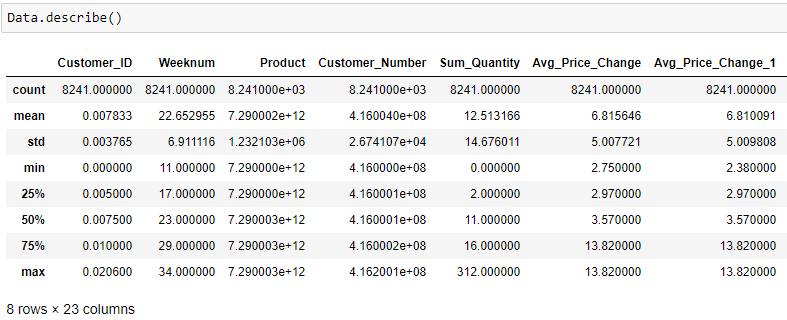
## Annexe 2 – Code description

Below we will explain in high level the code we wrote for the project, although we used two datasets and at the end compared the results, both 75 products and three products code are mostly similar.

In this annexe, I will explain the 3Product code, available on git (final-project-CSV\_v7\_3Products)

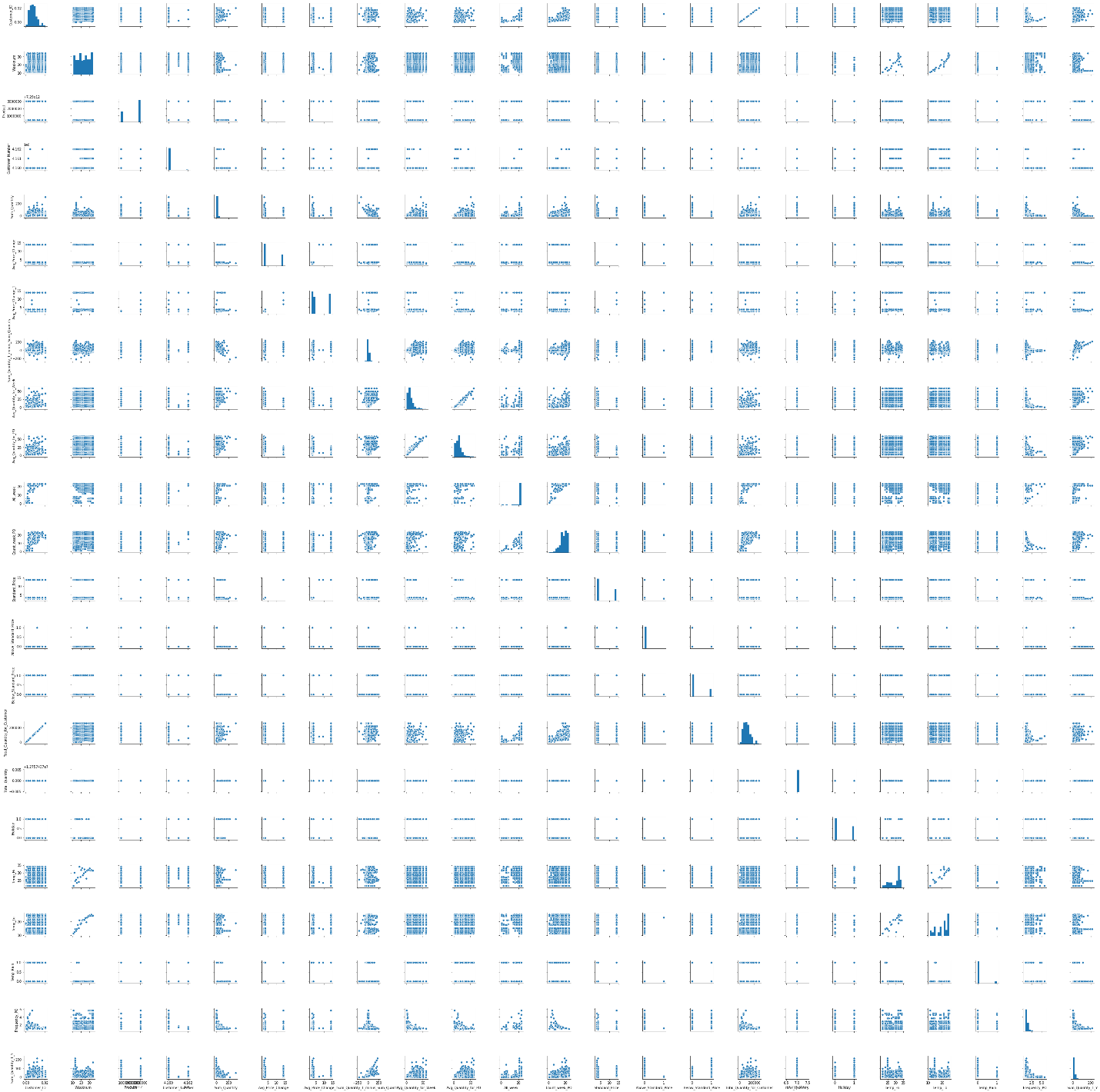
### Step 1 – Data retrieval

The data source is 3Products CSV dataset, from the description you can learn about the dataset column and values



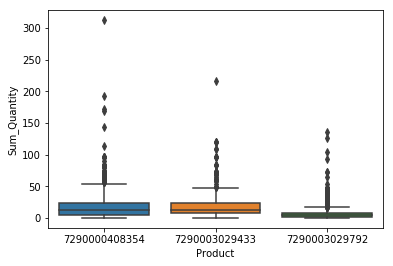
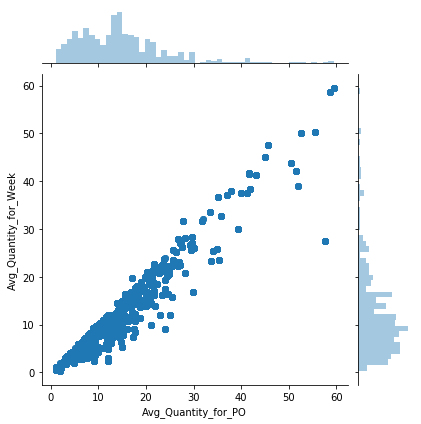
Step 2 – Pair plot (EDA)

All 23 parameters are presented in a paired plot allowing us to look for patterns in the data



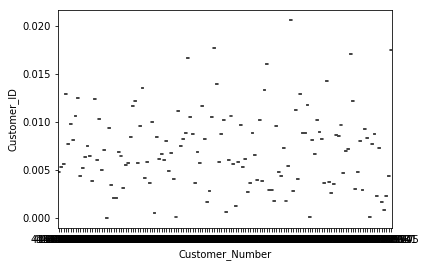
### Step 3 – Box and join plots

A few box plots show us how the data is distributed, in this example since we have only three products it can be seen more clearly than in the 75

### Step 4 – Client ID, Size

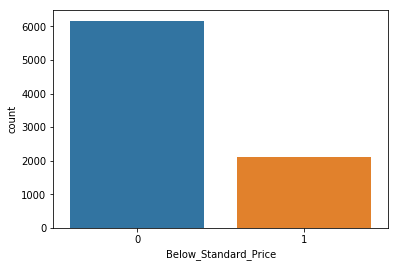
The identification of each franchisee was calculated according to its size, in the perspective of quantities, it allows us to have a meaningful number in the client\_id instead a useless continues number that might be taking into account by the algorithm, by doing so the below presents the franchises based on size



### Step 5 - Mechkar lib

You can find the results of Mechkar in the git under Documents

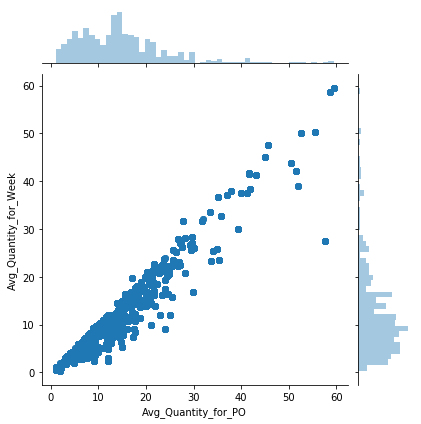
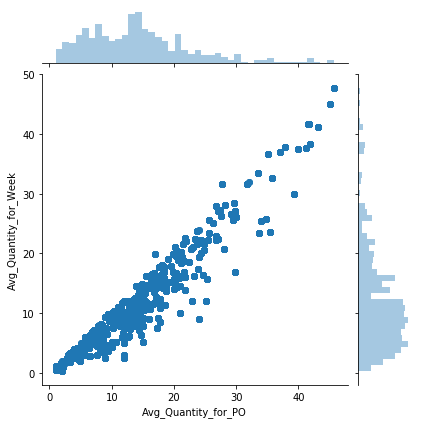
One of the interesting plots is *count vs below standard price*; here we can see how many order have been made while the products were on a special price



### Step 6 – Z score and Outliers (outliers detection)

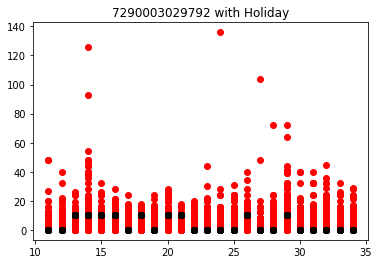
In this step we used Z score, average, mean and standard deviation to detect extraordinary values and based on that we have to define trash holds to a few parameters, those thresholds removed outliers from the dataset

For example, see the below join plots of average quantity per week vs average quantity for PO.

### Step 7 – Single product test

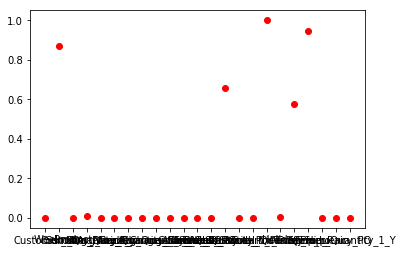
We have extracted a single product form the dataset to find out if we can learn something from it and how a single product effect the overall prediction. The extraction of a single product allowed us to see how the Holidays are reflected in the quantities of this product (the Holidays are in black, not-Holiday = 0, a week with a Holiday = 10)



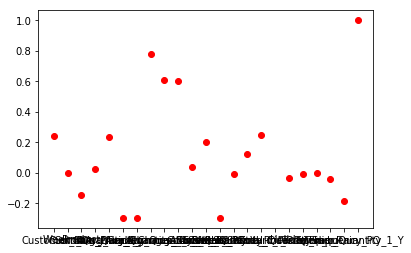
### Step 8 – P-Value and correlation (variable selection)

In this step we looked for a correlation between the different parameters, to be used in the following steps. To find it we used three methods Spearman, P value, Pearson correlation and Spearman.

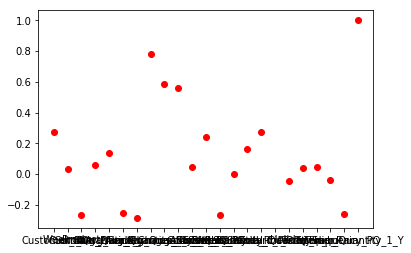
In the plot you can find the p-values of the different parameters:



While the Pearson correlation looks much different



And can easily be compared to Spearman



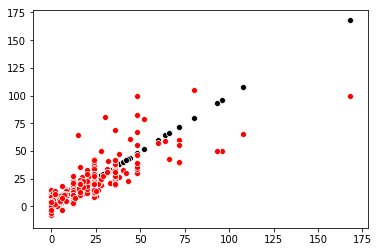
### Step 9- Train & Test Balance (train test preparation)

In this step, we looked for the number of orders (PO) of each product in the dataset. The intention was to balance the three products in a way that no one will dominate the dataset and will shift the others. We found that the dataset is fairly balanced with minor differences in the number of PO of the different products. We have decided to keep it as such without manual reduction or an artificial increase in the number of PO's in the dataset

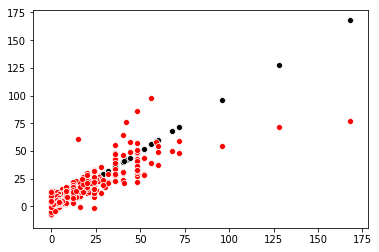
### Step 10 – Initial prediction calculation (model selection)

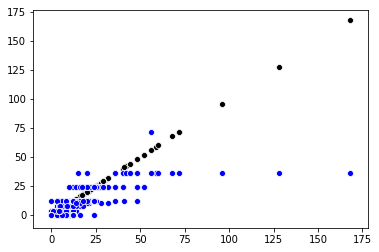
In this step, we have initiated the prediction algorithms while looking for the best result. We have tested the mean absolute error of both the train and the test data – to avoid overfitting.

* The best results are by using Lasso.



* Linear regression results were high.



* A good result has been achieved using Ada boost.
* 

### Step 10 – High correlation dataset (Variable selection)

In this step based on the results of the Spearman, p-value and Pearson correlation and cleaned the dataset accordingly, 11 parameters have been removed from the original dataset, and a new calculation of the MAE using the models has been done on the updated dataset.

Drop Data2 was a not successful try to manually get better results by manually choosing parameters to be removed from the original dataset

In most cases, the results were relatively the same or a bit better than the original dataset.

### Step 11 – Cross-validation (Fine Tune)

For this operation we keep on using the best algorithm we had and ran it using sklearn kfold, on all datasets, the best results we got was: Lasso cv=30, its MAE was 4.04

## Annexe 3 – High-level QA Protocol

|  |  |  |  |
| --- | --- | --- | --- |
| S/N | The operation | Required | notes |
| 1 | Data extraction | Required | Extract the data from the ERP system |
| 2 | Data accuracy | Required | Make sure that there are no animalities in the data |
| 3 | Data accuracy | Required | Make sure that the data is of the right product(s) |
| 4 | Data fulfilment | Required | Make sure that all fields are full |
| 5 | Data fulfilment | Optional | Add additional internal data |
| 6 | Data fulfilment | Optional | Add external data |
| 7 | Upload | Required | Upload the data to the model and check its availability in it |
| 8 | Run | Required | Run the model on the data |
| 9 | Results test | Required | Check the results you get from the model and compare to previous results you got from it |

## Annex 4 – Data models, results comparison

|  |  |  |
| --- | --- | --- |
| Method | 75 product DB (Mean Absolute Error, Train and Test) | 3 Product DB (Mean Absolute Error, Train and Test) |
| Linear regression | MAE train 14.532  MAE test 14.782 | MAE train 4.011  MAE test 4.15 |
| SVM | MAE train 29.780  MAE test 30.460 | MAE train 9.001  MAE test 9.320 |
| Ada Boost | MAE train 22.974  MAE test 23.582 | MAE train 6.205  MAE test 6.543 |
| Bayes | MAE train 107.428  MAE test 108.478 | MAE train 28.225  MAE test 28.223 |
| Logistic regression | MAE train 29.779  MAE test 30.460 | MAE train 9.001  MAE test 9.320 |
| Ridge |  | MAE train 3.965  MAE test 4.191 |
| Lasso |  | MAE train 4.960  MAE test 4.189 |

Above are the results of the models running a single run on the full dataset:

1. Customer\_ID float64
2. Weeknum int64
3. Product int64
4. Customer\_Number int64
5. Sum\_Quantity int64
6. Avg\_Price\_Change float64
7. Avg\_Price\_Change\_1 float64
8. Sum\_Quantity\_1\_minus\_Sum\_Quantity int64
9. Avg\_Quantity\_for\_Week float64
10. Avg\_Quantity\_for\_PO float64
11. All\_week int64
12. Count\_week\_PO int64
13. Standard\_Price float64
14. Above\_Standard\_Price int64
15. Below\_Standard\_Price int64
16. Total\_Quantity\_for\_Customer int64
17. Total\_Quantity int64
18. Holiday int64
19. Temp\_Hi float64
20. Temp\_Lo float64
21. Temp\_Rain int64
22. Frequency\_PO float64
23. Sum\_Quantity\_1\_Y int64

Note: no overfitting was identified

## Annexe 5 – Outliers

|  |  |  |
| --- | --- | --- |
| Outlier | Condition | Action |
| Product 9433  Sun Quantity | Higher then 190 | Removed (peak caused by Pesach) |
| Product 8354  Sun Quantity | Higher then 190 | Removed (peak caused by Pesach) |
| Product 8354  Average quantity per PO | Higher then 55 | 48 lines removed (affecting the linearity, wide spared) |
| Product 9433  Average quantity per PO | Higher then 55 | 22 lines removed (affecting the visual linearity, wide spared) |
|  |  |  |

## Annexe 6 – Feature selection

Three data sets have been examined by two high accuracy models.

Dataset : **Data**

Customer\_ID float64

Weeknum int64

Product int64

Customer\_Number int64

Sum\_Quantity int64

Avg\_Price\_Change float64

Avg\_Price\_Change\_1 float64

Sum\_Quantity\_1\_minus\_Sum\_Quantity int64

Avg\_Quantity\_for\_Week float64

Avg\_Quantity\_for\_PO float64

All\_week int64

Count\_week\_PO int64

Standard\_Price float64

Above\_Standard\_Price int64

Below\_Standard\_Price int64

Total\_Quantity\_for\_Customer int64

Total\_Quantity int64

Holiday int64

Temp\_Hi float64

Temp\_Lo float64

Temp\_Rain int64

Frequency\_PO float64

Sum\_Quantity\_1\_Y int64

dtype: object

Dataset: **DropData**

Customer\_ID float64

Product int64

Sum\_Quantity int64

Avg\_Price\_Change float64

Avg\_Price\_Change\_1 float64

Sum\_Quantity\_1\_minus\_Sum\_Quantity int64

Avg\_Quantity\_for\_Week float64

Avg\_Quantity\_for\_PO float64

Count\_week\_PO int64

Standard\_Price float64

Total\_Quantity\_for\_Customer int64

Sum\_Quantity\_1\_Y int64

dtype: object

and dataset: **DropData2**

Customer\_ID float64

Product int64

Sum\_Quantity int64

Avg\_Price\_Change float64

Avg\_Price\_Change\_1 float64

Sum\_Quantity\_1\_minus\_Sum\_Quantity int64

Avg\_Quantity\_for\_Week float64

Count\_week\_PO int64

Total\_Quantity int64

Sum\_Quantity\_1\_Y int64

dtype: object

While applying Linear regression and Adaboost models, the DropData dataset (they have the high correlation parameters) provides us with the best results.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Data | DropData | DropData2 |
| Adaboost | MAE: 6.54 | MAE: 5.84 | MAE: 7.34 |
| Linear reg | MAE: 4.14 | MAE: 4.1 | MAE: 4.12 |
|  |  |  |  |

Note: those are single run results, to validate those refouls we used Cross-validation applying on more methods.

## Annexe 7 - Cross-Validation

(MAE scoring)

In our perspective, this is the most important test, and therefore it has been performed using the three top models: Linear regression, Lasso and Ridge, in parallel it validates that DropData is the most suitable dataset.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CV\Dataset | 3 product Cross Validation[[10]](#footnote-10) | 3 product Cross Predict[[11]](#footnote-11) | 3 product CV of Drop data[[12]](#footnote-12) | 3 product CV of Drop data 2[[13]](#footnote-13) | 3 product  Lasso on Dropdata | 3 product  Ridge on Dropdata | 75 product Cross Validation[[14]](#footnote-14) |
| 2 | 7.684 | 16.699 | 7.719 | 102770 | 7069 | 166405 | 1044 |
| 5 | 4.216 | 12.756 | 4.187 | 4.258 | 4.082 | 4.127 | 7609 |
| 10 | 4.196 | 12.680 | 4.160 | 4.230 | 4.063 | 4.084 | 4587 |
| 15 | 4.197 | 12.703 | 4.161 | 4.235 | 4.046 | 4.053 | 3439 |
| 20 | 4.193 | 12.712 | 4.154 | 4.224 | 4.050 | 4.066 | 14.950 |
| 25 | 4.180 | 12.704 | 4.141 | 4.214 | 4.049 | 4.060 | 14.954 |
| 30 | 4.184 | 12.698 | 4.156 | 4.217 | 4.043 | 4.050 | 15.068 |
| 40 | 4.191 | 12.710 | 4.153 | 4.221 | 4.049 | 4.056 | 14.940 |
| 50 | 4.181 | 12.705 | 4.142 | 4.212 | 4.043 | 4.051 | 14.851 |

The cross-validation and cross predict in the table above have been done using Linear Regression, we added to it Lasso and Ridge as those are the most accurate.

We have also cross-validated using Ada boost however its results ware poorer.

The scoring of all are: Mean Absolute Error

1. <https://towardsdatascience.com/disruption-in-retail-ai-machine-learning-big-data-7e9687f69b8f> [↑](#footnote-ref-1)
2. An order can include a few lines, each line is an order of a single product [↑](#footnote-ref-2)
3. In it unique products are being sale [↑](#footnote-ref-3)
4. The summer holiday is define in low traffic. [↑](#footnote-ref-4)
5. See tables in Github <https://github.com/eyalfarkash/TCDS-Final.git> under Data [↑](#footnote-ref-5)
6. A test should be performed since CPU cost less but may take more time, while GPU is faster and more expansive [↑](#footnote-ref-6)
7. See at "Train&Test Balance" in "final-project-CSV\_v6\_3Products" [↑](#footnote-ref-7)
8. We have performed 5 tests "playing" with different parameters and group of parameters, all of them, results in lower accuracy comparing to "DropData", One is presented in the code. [↑](#footnote-ref-8)
9. Orders in the retail industry are based on packaging, and therefore the ordered quantity is round up to package size. [↑](#footnote-ref-9)
10. Running on the full dataset (Data), see more in section: Results\Feature Selection [↑](#footnote-ref-10)
11. Running on the full dataset (Data) see more in section: Results\Feature Selection [↑](#footnote-ref-11)
12. Running on the reduced dataset (DropData) see more in section: Results\Feature Selection [↑](#footnote-ref-12)
13. Running on the reduced manually dataset (DropData2) see more in section: Results\Feature Selection [↑](#footnote-ref-13)
14. Running on the full dataset (Data), see more in section: Results\Feature Selection [↑](#footnote-ref-14)