# Sales data to target variables

March 8, 2018

## 1 Sales data to target variables

### 1.1 Author: Gregorio Ferreira - ferreiradesajg@gmail.com

#### 1.1.1 Document description:

This notebook, contains the steps followed to pre-process and extract the target variables to train the models considering the sales history of each of the stores under consideration.

```
In [1]: import pandas as pd
        import numpy as np
        from scipy.stats import trim_mean
        import os
        os.chdir('XXX - YOUR PATH HERE - XXX')

        mydata = pd.read_csv("./data/sales_granular.csv",
                             index_col=0)

        ## looking for duplicates
        test_dup = mydata[mydata.index.duplicated(keep=False)]

        print('The store codes that are repeated wihtihn the original dataset is/are:\n')
        print(mydata.index.get_duplicates())
```

```
The store codes that are repeated wihtihn the original dataset is/are:

[11028]
```

```
In [2]: ## Dropping the duplicates and reshaping the data into a tidy data frame:
        ### Each feature represents store
        ### Each row is an observation of the number of sales.

        mydata = mydata.drop_duplicates(keep='first') ## keep='first' just for verbose

        mydata_t = mydata.T

        mydata_t.index = pd.to_datetime(mydata_t.index.values,
```

```
                format='%m/%d/%y %H:%M', errors='raise')

        print('The shape of sales data:\n')
        print(mydata.shape)
```

```
The shape of sales data:

(903, 11936)
```

## 1.2 List of assumptions

- The analysis need to be done in less than 15h.
- The size/type of the stores doesnt influence in the customer decision.
- The inventory levels/capacity are elastic. The distribution/availability of the product is 100% ideal. (There as many products available as necessary)
- The amount of revenue generated is the same for all the stores.
- If you travel less, you can give discounts to specific customers...
- Since the amount of sales is an absolute quantity:
- The total quantity is independent of the type of product.
- All the products, including competitors products, have the same visibility/positioning in all the stores
- The quantity is independent from the prices of all the products across all the stores and from the competitors prices
- Other brands positioning/marketing-campaigns does not affect the sales of the product
- The price fluctuation, if any, does not affect the consumption of the product.
- The surroundings and consumption
- The surroundings haven't changed during the period of time of this data.

```
In [3]: ## Resampling per months as the sum of the daily records.

        df_monthly = mydata_t.resample('M').sum()

        cols = list(df_monthly.columns)[1:]
        stores_sales = []
```

## 1.3 List of target variables

### 1.3.1 Continous targets:

- 10% Trimmed mean
- 20% Trimmed mean
- Sales average
- Minimum sales
- Maximum sales

```
In [4]: ## Continuous target variables
        for col in cols:
            stores_sales.append([col,
```

```
                        df_monthly[col].dropna().count(),
                        trim_mean(df_monthly[col].dropna(), 0.1),
                        trim_mean(df_monthly[col].dropna(), 0.2),
                        df_monthly[col].mean(),
                        df_monthly[col].min(),
                        df_monthly[col].max()])

    labels = ['store_code', 'count_non_na', 'trim_mean_01',
              'trim_mean_02', 'mean', 'min', 'max']

    stores_sales = pd.DataFrame(stores_sales, columns=labels)
```

### 1.3.2 Descrete targets:

The discretization is done as the quantiles of prob 0,2/0,4/0,6/0,8 to create five categories bottom/low/standard/high/top from each of the continuous targets. These are part of the assumptions to clusters the stores into five buckets.

- 10% Trimmed mean_class
- 20% Trimmed mean_class
- Sales average_class
- Minimum sales_class
- Maximum sales_class

In [5]:
```
## Creating quantiles classes
cols = list(stores_sales.columns)[1:]
col = cols[1]
for col in cols:

    stores_sales[col+'_class'] = np.nan

    myq02 = stores_sales[col].quantile(0.2)
    stores_sales.loc[stores_sales[col] < myq02, col+'_class'] = 'bottom'

    myq04 = stores_sales[col].quantile(0.4)
    stores_sales.loc[(stores_sales[col] >= myq02) &
                     (stores_sales[col] < myq04), col+'_class'] = 'low'

    myq06 = stores_sales[col].quantile(0.6)
    stores_sales.loc[(stores_sales[col] >= myq04) &
                     (stores_sales[col] < myq06), col+'_class'] = 'standard'

    myq08 = stores_sales[col].quantile(0.8)
    stores_sales.loc[(stores_sales[col] >= myq06) &
                     (stores_sales[col] < myq08), col+'_class'] = 'high'

    stores_sales.loc[stores_sales[col] >= myq08, col+'_class'] = 'top'
```

```
stores_sales.to_csv("./result dataset/stores_sales.csv", sep=',', encoding='utf-8', inc
```

In [ ]: