A Work Project, presented as part of the requirements for the Award of a Master Degree in Management from the NOVA – School of Business and Economics

SALES MODEL:

A PRELIMINARY APPROACH AND METHODOLOGY

MARIA TERESA HENRIQUES DE SOUSA LOPES E PÉREZ

34140

A Project carried out on the Master in Management Program, under the supervision of:

Professor Ana Amaro

03/01/2020

Table of contents

Abstract

This work project aims at supporting managers in explaining and predicting sales of specific products based on a devised methodology. Product sales time series were analysed and processed in order to select the best model type: explanatory models (through ordinary least squares method), univariate models (Box Jenkins methodology) or dynamic models mixing up the two previous approaches. An automatic procedure to put the methodology in practice was implemented using Python, due to the huge amount of product sales to be modelled. The process was tested using data from more than 1500 products from Beiersdorf Lisbon. For the sake of confidentiality, the names of the products were modified. The most accurate models are described and analyzed.

Keywords: Regression, ARIMA, Box Jenkins, Sales model

This work used infrastructure and resources funded by Fundação para a Ciência e a Tecnologia (UID/ECO/00124/2013, UID/ECO/00124/2019 and Social Sciences DataLab, Project 22209), POR Lisboa (LISBOA-01-0145-FEDER-007722 and Social Sciences DataLab, Project 22209) and POR Norte (Social Sciences DataLab, Project 22209).

Introduction

The importance of a sales explanatory model

Explanatory models have always been essential tools to business decision makers. Living in a world in constant change due to the speed at which information flows represents a difficult challenge to managers that are forced to make decisions on a day to day basis and therefore to adjust their core strategies. Simultaneously these managers are pressed to increase revenues and to be competitive against more flexible companies with a worldwide scale.

The ability to explain the behavior of sales through external factors can contribute to a better understanding of the right strategy to follow. Sales models support managers with information on what action to take if some uncontrollable parameters change. For example, there are some products whose sales to client A will drop when another client B opens a new store (mind that Beiersdorf assumes retailers as clients and not the final customer). On the other hand, sales to client A increase when products are in promotion price. Therefore, when Beiersdorf learns that client B is going to open a new store, it can prepare to set this product in promotion price, to offset the negative impact on the products sales. This is important to the extent that the model can support managers when making strategic decisions across different areas of the company - advertising, point of sale activation, promotion prices, financial investments, among others.

Additionally, explanatory models, when sufficiently accurate, can be used to make predictions. Having a sales forecasting model is an increasing helpful tool to predict the business performance in the upcoming months. Moreover, not only can it support plans of action and growth across the aforementioned areas of the company but also it assists companies supply in meeting the changing demand.

Company Overview

Beiersdorf is a German multinational company founded in 1882. Today it has 20,000 employees and more than 160 affiliates worldwide (Beiersdorf, 2019). The company is divided into two segments: **Consumer Business Segment** and **Tesa Business Segment**. The **Consumer Business segment** main focus is on skin and body care markets. Their international brand portfolio remains relevant and specific needs and wishes from customers are satisfied through constant innovations and by staying close to the consumers. **Tesa Business segment** has worked as an independent unit since 2001, offering superior and reliable technology within high quality products. Focused on innovation, Tesa is one of the world's leading manufacturers of self-adhesive product solutions for industrial customers and consumers.

The main goal and focus of Beiersdorf is to make people feel good in their skin. Aiming to become the number one skin care company in the world, Beiersdorf caters every sort of customer need and operates in different markets - mass market, dermo cosmetics, and premium. In the current economic context, competition is particularly significant, which demands constant investments in Research and Development (R&D) and improvement of new and existing products as well as processes. In many countries, its brands and specially NIVEA are perceived to be local because of the development of skin care products according to the country's specific needs. For this purpose, besides the R&D Center in Hamburg, Beiersdorf has Regional Development Laboratories in Mexico, China and India.

Besides **NIVEA**, Beiersdorf's most important brand (Beiersdorf, 2019), the other brands present in Portugal are **Eucerin**, **Hansaplast**, **Labello**, **Atrix**, **Fuss Frisch** and **Harmony**. For the context of this thesis, NIVEA will be the only brand apprised.

Literature Review

This section presents a summary of the research conducted not only about explanatory and sales forecasting models, but also on how to implement these methods in Python. Traditional forecasting methods include not only multiple linear regression but also time series models,

which use historical data to forecast and explain the behavior of the dependent variable. Among the time series methods, such as the Naive method, average method, exponential smoothing, Holt's linear and exponential trend method, damped or seasonal trend methods, the most common ones are the moving averages, ARMA (Autoregressive Moving Average), and ARIMA (Autoregressive Integrated Moving Average). These have been previously used to forecast a wide range of matters. In a study from Bentley College (Weisang and Awazu, 2014), they proposed the application of an ARIMA to forecast the USD/EUR exchange rate. Whereas in a paper from the School of Science of Xi'dian University the same model was used to predict real-time rain-induced attenuation (Radio Science, 2013). Furthermore, literature on forecasting methods such as *Forecasting: principles and practice* (Hyndman and Athanasopoulos, 2014), *Introduction to time series and forecasting* (Brockwell and Davis, 2016) and *A course in time series analysis* (Peña, Tiao and Tsay 2001) served as a drive to use this method, since they describe the ARIMA as a simple and powerful model. In recent years, researchers have been developing systems to forecast sales through machine learning, as it was done at Istanbul University (Kilimci, Akyuz, Uysal, Akyokus, Uysal, Atak Bulbul and Ekmis 2019). To the extent of this thesis, a half-automatic methodology was developed in Python based on several articles published by Raphael Bubolz Larrosa (Towards Data Science, 2019), Sangarshanan (Towards Data Science, 2018), Kostas Hatalis (Data Science Central, 2018). The code is used as a tool to help modeling all product sales and therefore it requires the user to analyze graphs as well as statistical tests. The implementation of the code required constant assistance of online forums such as Stack overflow, GitHub and from the Pyhton libraries documentations.

Data description and model proposals

Beiersdorf company provided a data set with 1668 time series for more than 500 different products sales for three different clients. For each product category information was provided on the promo intensity (percentage of sales in promotion price), market penetration and market share of NIVEA and its main competitors. The media plan for each product along the past five years was also made available. Additionally, information was given about the number of stores

of each client for the last five years. External data on social and economic factors in Portugal (Pordata 2018) were collected, based on research performed on the company's financial reports (Beiersdorf, 2019) and industry analysis (Essays UK, 2018). These were the following: Consumption by households in the economic territory as % of GDP, GDP growth rate, Unemployment rate per gender, average monthly wage per gender and guests in tourist accommodations per 100 inhabitants.

Three different types of models were proposed to define product sales $\hat{S}_{pst}$ of a given product p $(p = 0, \ldots, \bar{P})$ in month t $(t = 0, \ldots, \bar{T})$ to client s $(s = 1, 2, 3)$, as will be explained further. Foremost, the following variables were defined as:

PI_{cbt}, a continuous variable that represents the promo intensity per product category c, per brand b, per month t

MP_{cbsw}, a continuous variable that represents the market penetration per product category c, per brand b, per client s per trimester w

MS_{cbt}, a continuous variable that represents the market share per product category c, per brand b, per month t

A_{pt}, a binary variable 1 if product p is on TV advertisement in month t and 0 otherwise.

C_{skt}, a discrete variable representing the number of stores of client s, of type k in month t

E_{iy}, a continuous variable representing external factor i in year y.

The significance of these variables on the sales of each product was determined by developing a Multiple Regression model. For the sake of simplicity, a variable F will be used to represent the sum of all the factors previously presented, that is $F = PI + MP + MS + A + C + E$. The product sales that can be modeled through a linear regression model, are given by the following equation:

$$\hat{S}_j = \beta_0 + \sum_{j=1}^{j=\bar{J}} \beta_j(F_j) + \varepsilon_j, \bar{J} = \bar{P} \times \bar{T} \tag{1}$$

Where β_0 is the y-intercept and β_q is the regression coeficient of each F_j. The determination of the values for the coeficients β_j is done through the ordinary least squares (OLS) method. The best-fit model will be the one that minimizes the sum of squared differences between the observed and fitted values of product sales. Therefore, the coefficients are given by $\beta_j = (S_j - \hat{S}_j)^2$. Although the OLS gives the best-fit model, there is still the need to verify its overall significance and check if no redundant parameters are being considered. For these purposes an F-test must be performed and t-tests for each variable are also required. When there is not enough evidence to reject the null hypothesis of a t-test, it means that either this parameter is not required to explain the sales of the product or that it is correlated to one of the other variables. In order to analyze which of the latter situations is the case, a computation of the variance inflation factor (VIF) should take place. Then, the parameter with the highest VIF is removed from the model.

In case that the variables above have no added value when it comes to the description of the product sales, an univariate model can be applied. An auto regressive integrated moving average method (ARIMA) aims to describe the dependent variable's current behavior through linear relationships with their past values. An ARIMA model is composed of two parts. First, there is the integrated (I) factor (d), $d = 0, 1, 2$ which represents the order of differencing required to make the series stationary. Since this type of model works as a linear regression that takes its own previous values as regressors, these need to be independent from each other, which is not the case in non-stationary time series. The second constituent of the ARIMA is the ARMA, which can in turn be divided into two components, the AR and MA. The autoregressive (AR) element (p) expresses the correlation between the sales current value and one or more of its past values. Meaning that, for instance, p is 1 at month t, then the sales are correlated to its value at month $t - 1$. On the other hand, the moving average (MA) component q captures the duration of the impact that a shock has on the time series. In particular, if q is 1 in month t it means that the current value of the product sales time series is correlated with the error of month $t - 1$. The values of p and q can be estimated through the analysis of the time series autocorrelation function (ACF) and partial autocorrelation function (PACF). The value of

p can be found in a PACF correlogram: the first lags that are significant (above the sifnicance line) will be the order of the AR. The same can be done to the value of q using the ACF plot, which shows the number of MA terms required to remove any autocorrelation from the series (David Abugaber, 2019).

Subsequently, the product sales will be defined as follows.

$$d.\,\hat{S}_{pt} = c + \sum_{i=0}^{i=\bar{P}} \phi_i s_{t-i} + \sum_{k=0}^{k=\bar{Q}} \theta_k \varepsilon_{t-k} \tag{2}$$

Where $d.$ is the differencing operator. Moreover, $\sum_{i=0}^{i=\bar{P}} \phi_i y_{t-i}$ is the AR polynomial, and $\theta(B)$ is the MA polynomial and can be defined as $\sum_{k=0}^{k=\bar{Q}} \theta_k \varepsilon_{t-k}$. $\bar{P}$ and $\bar{Q}$ are the orders of the autoregressive and the moving average components, respectively.

Additionally, when the time series are seasonal, there are other components that can be included in the ARIMA model to represent this, becoming a seasonal auto regressive integrated moving average (SARIMA) model. The additional seasonal terms $(P, D, Q)_m$ are similar to the non-seasonal parts of the model, although in this case the values backshift a seasonal period instead of an immediate period before. For instance, a $SARIMA(0,0,0)(1,0,0)_{12}$ means that the sales of the current month are correlated to the sales of one year (12 months) ago. Thus, the seasonal part of the model can simply be added to the ARIMA equation, as follows.

$$d.\,D.\,\hat{S}_{pt} = c + \sum_{i=0}^{i=\bar{P}} \phi_i s_{t-i} + \sum_{k=0}^{k=\bar{Q}} \theta_k \varepsilon_{t-k} + \sum_{l=0}^{l=P} \Phi_l s_{t-l} + \sum_{n=0}^{n=Q} \Theta_n \varepsilon_{t-n} \tag{3}$$

Where $\sum_{l=0}^{l=P} \Phi_l s_{t-l} + \sum_{n=0}^{n=Q} \Theta_n \varepsilon_{t-n}$ is the seasonal polynomial. P and Q are the orders of the seasonal auto regressive and seasonal moving average, respectively. Whereas D is the differencing order required to make the seasonal component stationary. Note that $d + D < 2$.

Despite the considerable quality of both the multilple regression and the ARIMA univariate models, when considered independently, it is possible that significant information is being ignored. The multiple regression does not take into acount historical values, whereas the ARIMA univariate does not consider external factors. A simple approach to tackle this issue is

to combine the two methods in a dynamic model. In practice, the methodology to estimate the ARIMA/SARIMA factors values is the same as for the univariate model. However, once explanatory variables are added, some past values might become redundant, or even not significant at all. Hence, t-tests need to be performed to all regressors to make sure that both explanatory and past values are all adding value to the model. Thus, the product sales which can only be described by a dynamic model will be given by the following equation.

$$d.D.\hat{S}_{pt} = c + \sum_{i=0}^{i=\bar{P}} \phi_i s_{t-i} + \sum_{k=0}^{k=\bar{Q}} \theta_k \varepsilon_{t-k} + \sum_{l=0}^{l=P} \Phi_l s_{t-l} + \sum_{n=0}^{n=Q} \Theta_n \varepsilon_{t-n} \sum_{j=1}^{j=\bar{J}} \beta_j(F_j) \quad (4)$$

Methodology

A preliminary graphical analysis was performed along with expertise discussions with the company decision makers. At this stage, some data were removed from the dataset. The database provided included products that have been already discontinued, and obviously there is no interest in analyzing such products. Likewise, products which were introduced in the Portuguese market less than 5 years ago were discarded since there is not enough information to perform any type of analysis. The data cleaning resulted in a dataset with over 300 products (over 100 products per client). Due to this large number of products, after a preliminary analysis to assess the likelihood of generating new information with the available methods, the following methodology was implemented using Python:

1. Product time series graphical analysis. Should it be absolutely clear from the graph of the time series that this product is seasonal (Fig 2), move on directly to a dynamic model, step 6. Otherwise (Fig 1), follow step 2.

Fig. 1. Time series plot that does not appear to be seasonal nor stationary

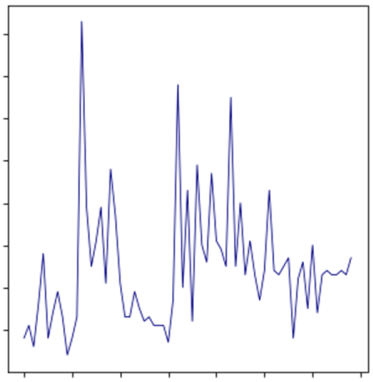
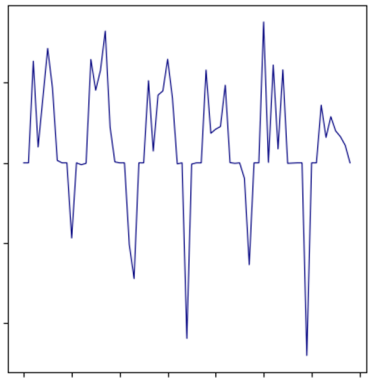


Fig. 2. Time series plot that clearly appears to be seasonal



2. Build four sets of data for the product in question, which consider each possible transformation to the variables of the model: Lin-Lin; Lin-Log; Log-Lin; Log-Log (DEV, 2019). This was done as an attempt to transform highly skewed data into a more normalized variable, in order to make it better interpretable: using a log base of 10, a change of one on the log scale is equivalent to a product of ten on the original data.
3. Explanatory model approach.

   3.1. For each product sales establish explanatory linear models, starting with the most correlated explanatory variable and adding new ones with the criteria that maximizes the adjusted r-squared (Hyndman and Athanasopoulos 2014, 3.2).

   3.2. Analise the residuals in order to ensure that the assumptions of the regression model are satisfied. Evaluate if there is a pattern in the scatter plot of the residuals against the fitted values and assess if the residuals follow an approximately normal distribution (Hyndman and Athanasopoulos 2014, 5.3).

   3.3. Perform an F-test to check the overall significance of the model.

   3.4. Perform t-tests to check the individual significance of the explanatory variables.

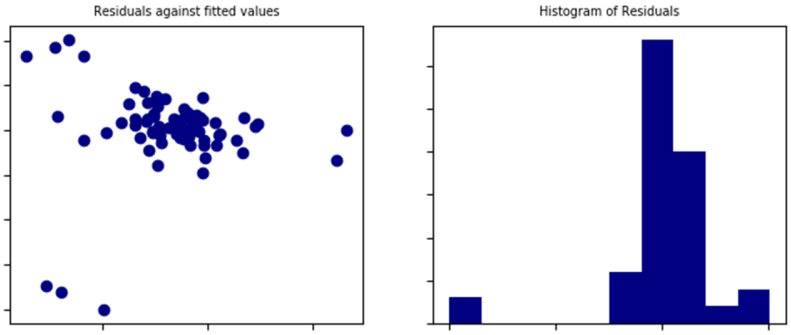
   3.5. Compute the Variance Inflating Factor (VIF) to check for multicollinearity. Remove the parameter with the highest VIF and for which the null hypothesis was rejected in the t-test. Start over until all the parameters are significant.
4. Assessing the accuracy of the model to understand if a different one should be used.

If the sales product linear model:

4.1. is non significant or r-squared<0.4 (Robert Nau, 2019) an ARIMA/SARIMA approach (5.) shall be followed.

4.2. seems to be significant but the residuals present an autocorrelated pattern (Fig. 3) a dynamic model approach (6.) shall be followed.

Fig. 3. Graphs showing that the residuals are autocorrelated and do not follow a normal distribution with mean zero.

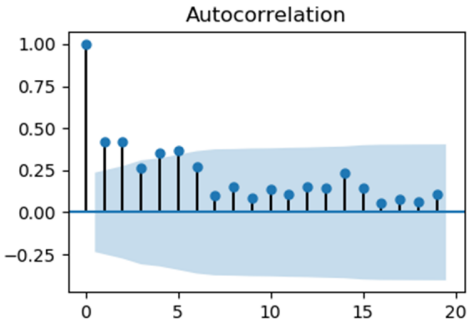


4.3. or if the time series graph clearly shows seasonality (Fig 2), a dynamic approach (6.) shall be followed.

5. Univariate model approach

5.1. Correlogram analysis (Fig. 4) and Augmented Dickey Fuller test to evaluate stationarity of the dependent variable (Towards Data Science, 2018). The series need to be stationary to apply an auto regressive method, because this is a linear regression model that uses past values of the dependent variable as explanatory variables (Data Science Central, 2018). Therefore, they need to be uncorrelated and independent from each other, which happends only in series that are stationary. Hence, if the series is not stationary, compute a first difference (re run ADF test and if not stationary compute a second difference). The differencing order shall be the value of d for the $ARIMA(p, d, q)$ parameters.

Fig. 4. A correlogram of non-stationary time series

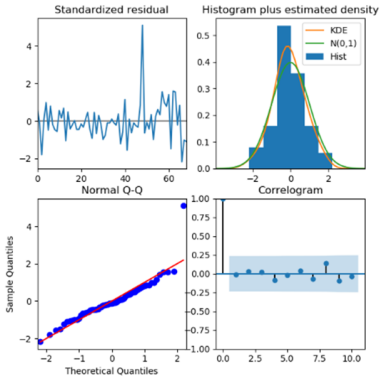


5.2. Correlogram anlaysis to determine the values of p and q from $ARIMA(p, d, q)$: p will be the number of lags of Y used as regressors. Whereas q is the number of lags of the error term that should go in the model.

5.3. Experiment different combinations of orders for the AR and MA terms, having as maximum values for p and q the ones found in the previous step. The combination that gives the least AIC (Hyndman and Athanasopoulos 2014, 8.6) and has only significant parameters is the best model.

5.4. Anlise the residuals (Fig. 5) to ensure the model obtained is a good fit (Bizstats.ai, 2019). The residuals should follow a normal distribution with mean zero and have a uniform distribution. Moreover, the ordered distribution of residuals should follow the linear trend of the samples taken from a standard normal distribution, which reinforces the normality of the residuals. Lastly, the residuals should appear to be white noise, which can be assessed through a correlogram.

Fig. 5. Residual analysis of a good fit model



6. Dynamic model approach

    6.1. On this case exactly the same steps as for the univariate ARIMA model should be followed, only now instead of being univariate, this model also has exogenous variables (Towards Data Science, 2018).

    6.2. Perform t-tests to make sure that all the parameters are significant. If not, remove the non significant parameters from the model and repeat the residual analysis (Fig 5).

7. Interpretation of results

Experimental Results

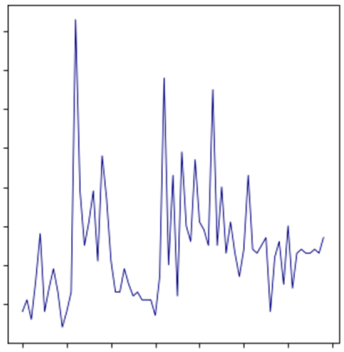
Experimental findings disclose that the methodology above has a good level of accuracy for many Beiersdorf products. It was applied to all the products of one particular client. Before going into further detail, the overall analysis shows that 28% of the products were modeled through a multiple regression, 12% were explained by an ARIMA univariate approach, 8% by a SARIMA univariate method and 46% of the products had its sales described using a dynamic model. Despite the good results obtained, this model did not present any accurate results for 6% of the products.

With the purpose of illustrating the proposed methodology, a few applications of the algorithm are demonstrated bellow. To keep the confidentiality of the data, the product names were replaced by numbers and the names of the variables are treated using the characterization previously presented.

Product 24

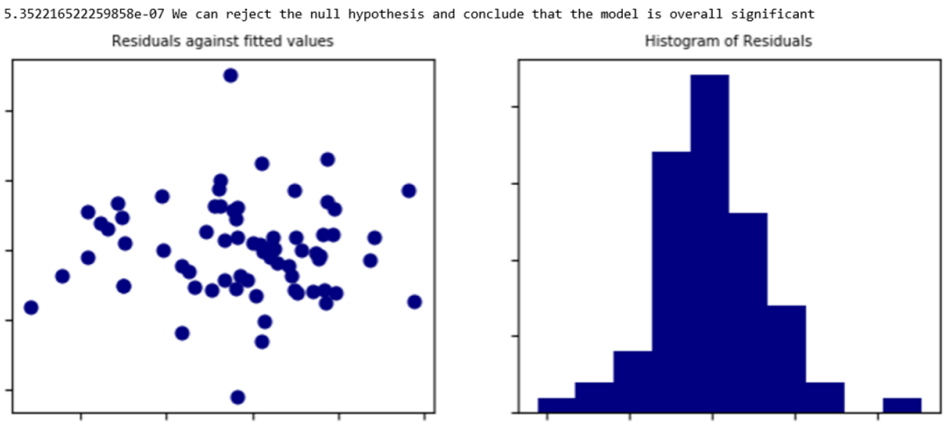
Product 24 time series (Fig 6), does not seem to be stationary nor seasonal.

Fig. 6. Product 24 time series plot



Log-Lin was the best-fit model resulting from applying the forward selecting strategy to choose significant regressors and making logarithmic transformations to the data. That is, a logarithm transformation was applied to the sales values. The model has an r-squared of 0.658, which is higher then the level of acceptance, suggesting that it seems to have significant parameters. A graphical analysis of the residuals shows that they satisfy the assumptions of the model, i.e. not only do they follow a normal distribution, but they also have no pattern when plotted against the fitted values (Fig 7).

Fig. 7. Residual analysis of product 24 best-fit Regression model and F-test for the overall significance of the model



An F-test (Fig 7), shows that the model is overall significant. However, it is not possible to reject the null hypothesis of the t-tests for some parameters, meaning that these may be

redundant. Through an analysis of the value inflation factors and t-tests, these non-significant regressors were removed from the model. Finally, *product 24* sales can be described as follows:

$$\begin{aligned} Log(\hat{S}_{24,2,t}) = -8.5515 + 8.8471MS_{2,3,t} + 209.728MP_{2,3,2} + 26.2855MP_{2,0,0} \quad (5) \\ -117.8003MP_{2,5,3} + 378.3133MP_{2,3,3} + 146.6778MP_{2,5,1} - 0.7727C_{1,1} \\ +0.0654C_{2,2} - 0.0286C_{1,3} - 0.05254C_{3,3} \end{aligned}$$

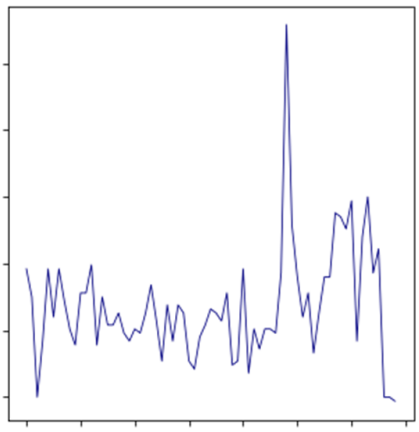
Since the final model has an r-squared of 0.602, it can be stated that 60.2% of the variation in the sales of *product 24* is due to changes in the following factors: *market share of brand 3* $(MS_{2,3,t})$, or in the *market penetration of category 2 by the market* $(MP_{2,0,0})$, or by brands *5* $(MP_{2,5,3})$ and *3* $(MP_{2,5,3})$ at *client 3*. It can also be due to changes in the *market penetration* by *brand 5 in client 1* $(MP_{2,5,1})$. Moreover, changes in the *number of stores 11* $(C_{2,2})$, *22* $(C_{2,2})$, *13* $(C_{1,3})$, or *33* $(C_{3,3})$ also contribute to explaining these 60.2% of sales variation.

In other words, if *client 1* opens a new store *type 3* $(C_{1,3})$, the sales of *product 24* at *client 2* will decrease by 0.286 units. In order to tackle this penalty in sales, NIVEA can engage in strategies to increase the *market penetration* of *category 2* $(M_{2,0,0})$, for instance. If the latter increases by 1%, then the sales of *product 24* at *client 2* will increase by 262.855 units. Furthermore, if the *market penetration* of *brand 5* in *client 3* $(MP_{2,5,3})$ increases 1%, then the sales of *product 24* drop 1178 units. However, an increase of 1% in the *market penetration* of *brand 3* $(MP_{2,3,3})$ in the same client boosts sales of *product 24* by 3783.133 units. Thus, if NIVEA foresees that *client 1* is going to open a *type 3* store, it can partner up with *brand 3* to either increase its *market penetration* at *client 3* $(MP_{2,3,3})$ or to increase their *market share* $(MS_{2,3,t})$. It should be stressed that the aforementioned strategy of increasing a competitor's *market share* (MS) might not be the best path to follow since it might have a negative impact on the sales of many other products, despite the positive contribution for the sales of *product 24* to *client 2*. Therefore, promoting a positive growth in the *market penetration* (MP) of a competitor brand in this particular category should be the best strategy, as it contributes to increase NIVEA's sales of *product 24* to *client 2*, without negatively impacting the sales of other products.

Product 76

Product 76 time series plot (Fig 8) shows a high variance, especially in recent periods, but there are no clear signs of stationarity or seasonality.

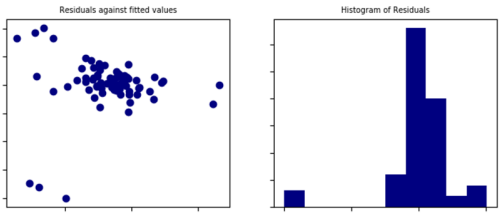
Fig. 8. Product 76 time series plot



Accordingly, logarithmic transformations were made to the data followed by the forward selecting strategy to choose the best regressors to describe this product sales. The resulting best-fit model had a low r-squared (0.248) and residuals that did not verify the assumptions of the regression (Fig 9).

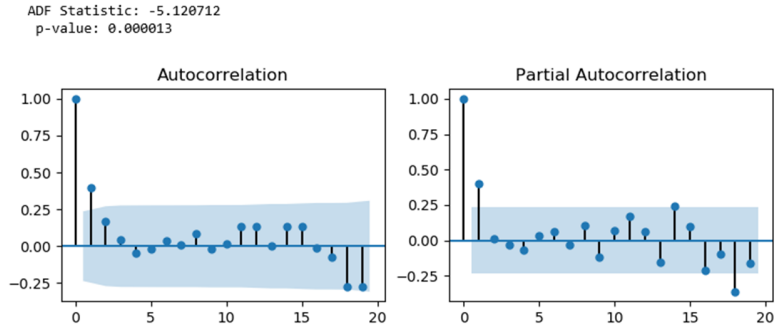
Fig. 9. Results analysis for product 76 best-fit Regression model

Dep. Variable:	logy	R-squared:	0.248
Model:	OLS	Adj. R-squared:	0.174
Method:	Least Squares	F-statistic:	3.344
Date:	Sun, 29 Dec 2019	Prob (F-statistic):	0.00649
Time:	15:41:25	Log-Likelihood:	-94.107
No. Observations:	68	AIC:	202.2
Df Residuals:	61	BIC:	217.8
Df Model:	6		
Covariance Type:	nonrobust		



Hence, an ARIMA/SARIMA univariate should be applied. The ADF test shows that the series seems to be stationarity (Fig 10). The same can be observed in the correlogram of autocorrelation and partial autocorrelation, together with the fact that the sales of this product are not seasonal. Hence, an $ARIMA(p, 0, q)$ method was pursued. The correlogram (Fig 10) suggests that the parameters p and q of the ARIMA are both at most, 1.

Fig. 10. Results of ADF test . ACF and PACF Correlograms for product 76

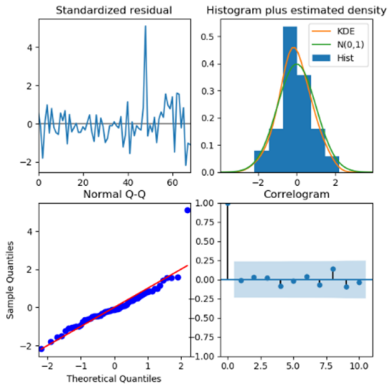


Through the tryout of several combinations of p and q values, the best-fit model is an $ARIMA(1,0,1)$. Consequently, the sales of *product 76* to client 2 can be explained by the following equation:

$$\hat{S}_{34,2,t} = 135.5775 + 0.41S_{34,2,t-1} \tag{6}$$

The residuals satisfy the assumptions of the model (Fig 11) since their variance is approximately uniformly distributed and the residuals seem to be normally distributed with mean 0. Additionally, the qq-plot shows that the ordered distribution of residuals falls almost perfectly on the linear trend of the samples taken from a standard normal distribution. Lastly, the correlogram of the residuals seems to be white noise.

Fig. 11. Residual analysis of product 76 sales best-fit model

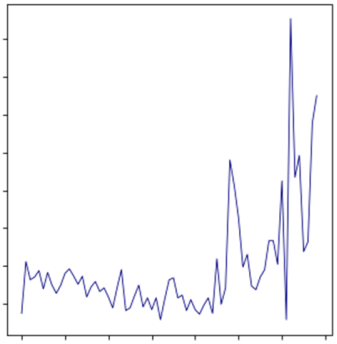


Thus, it can be concluded that the sales of product 34 will be predictable based on the value of the sales of the previous month. In other words, the current sales of product 34 are highly correlated (0.4) with the previous month sales.

Product 75

The time series plot (Fig 12) of *product 75* suggests no stationarity and no seasonality.

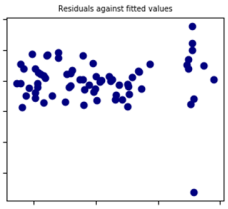
Fig. 12. Product 75 time series plot

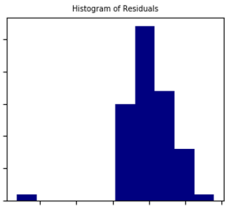


After making logarithmic transformations to the data, the forward selecting strategy to find the regressors that better explain the sales of *product 75* was applied. The best-fit regression model obtained had residuals that did not satisfy the assumptions of the regression (Fig 13) and appeared to have a high r-squared (0.624).

Fig. 13. Results analysis for product 75 best-fit Regression model

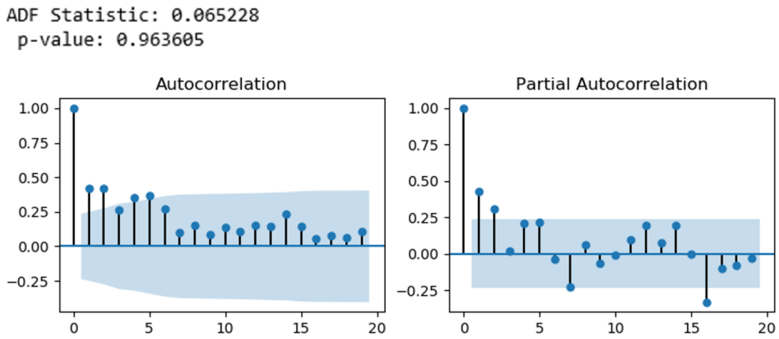
Dep. Variable:	logy	**R-squared:**	0.624
Model:	OLS	**Adj. R-squared:**	0.581
Method:	Least Squares	**F-statistic:**	14.46
Date:	Sun, 29 Dec 2019	**Prob (F-statistic):**	6.32e-11
Time:	18:02:24	**Log-Likelihood:**	-24.955
No. Observations:	69	**AIC:**	65.91
Df Residuals:	61	**BIC:**	83.78
Df Model:	7		
Covariance Type:	nonrobust		





Therefore, a dynamic approach should be followed. The ADF test (Fig 14) shows that the series are not stationary, so it needs to be differenced. Analyzing the correlogram (Fig 14), the values of p and q had to be be lower than 2. Thus, the model used was an $ARIMA(p, 1, q)$ with significant exogenous variables.

Fig. 14. Results of ADF test . ACF and PACF Correlograms for product 75

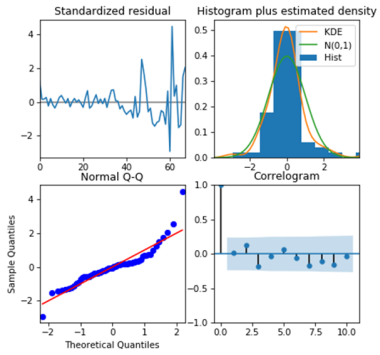


The best-fit model is an ARIMA(0,1,1) with the variable that represents the *promo intensity* of this product category (12) at *client 3* $(PI_{12,3})$. In other words, the sales of *product 75* to client 2 are given by the following equation:

$$d_1.\,\hat{S}_{75,2,t} = 37.5523 - 181.4459PI_{12,3} - \varepsilon_{t-1} \tag{7}$$

The variance of the residuals is approximately uniform and they seem to follow a normal distribution with mean 0. Additionally, the ordered distribution of residuals falls almost perfectly on the linear trend of the samples taken from a standard normal distribution. Their correlogram seems to be white noise (Fig 15).

Fig. 15. Residual analysis of product 75 sales best-fit model

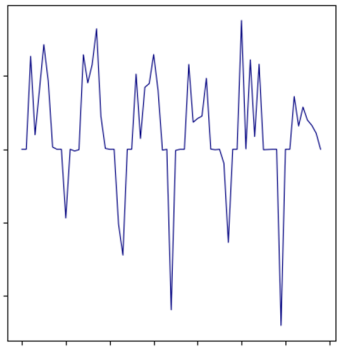


Therefore, the sales of *product 75* to client 2 can be explained by the *promo intensity* of *brand 3* in *category 12* $(PI_{12,3})$ and are negatively correlated with the error of the previous period (ε_{t-1}). If the *promo intensity* of *brand 3* $(PI_{12,3})$ declines 1%, the sales of *product 75* to client 2 increase approximately 181 units. With this information, NIVEA's strategy can be adapted to influencing *brand 3 promo intensity* to drop.

Product 90

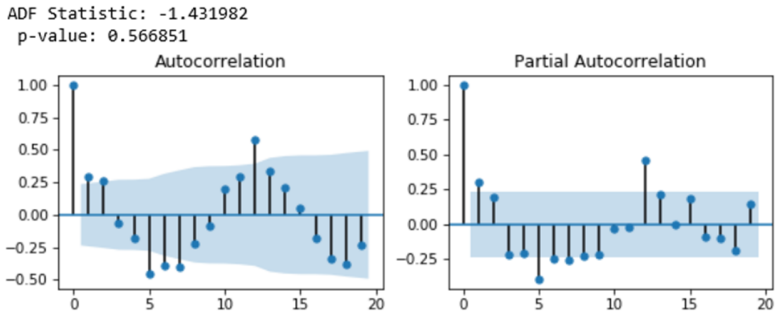
The time series plot of *product 90* clearly shows seasonality (Fig 16).

Fig. 16. Product 90 time series plot



Hence, a dynamic approach should be pursued. Both the ADF test and the correlogram (Fig 17) show that the series are stationary, thus no differencing is required. Additionally, the correlogram of autocorrelation reinforces the seasonality of this product's sales. The model used was therefore a $SARIMA(p,0,q)x(P,0,Q,12)$. Once again through the analysis of the autocorrelation and partial autocorrelation correlograms (Fig 17), it can be concluded that the maximum values for p, q, P and Q are 1, 2, 1 and 2, respectively.

Fig. 17. Results of ADF test . ACF and PACF Correlograms for product 90

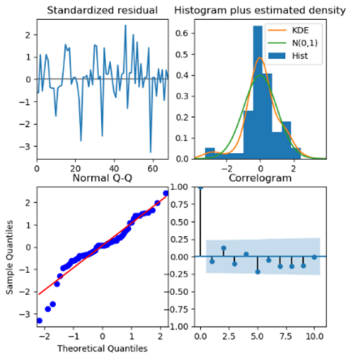


After observing all the possible combinations of orders of the SARMA parameters, the best-fit model is a $SARIMA(0,0,0)x(2,0,0,12)$. Hence, *product 90* sales can be explained by the equation bellow.

$$\hat{S}_{90,2,t} = 3.4704 + 2.951 \times 10^{-5} MP_{10,3,3} \\ +0.4767 S_{90,2,t-12} + 0.2849 S_{90,2,t-24} \tag{8}$$

The residuals (Fig 18) follow a normal distribution with mean 0 and its variance has an approximately uniform distribution around 0. Moreover, their correlogram is white noise and their ordered distribution falls almost perfectly on the linear trend of the samples taken from a standard normal distribution.

Fig. 18. Residual analysis of product 90 sales best-fit model



This model discloses that the sales of *product 90* are approximately 0.48 times greater than in the previous year and 0.28 times more than two years before. This means that the sales of *product 90* are predictable based on historical values. Furthermore, it can be stated that the *market penetration* of *brand 3* in *client 3* $(MP_{10,3,3})$ has a small, but positive impact on *product 90* sales. Thus, in order to increase the sales of product 90, NIVEA can partner up with brand 3 to increase their market penetration in this category and client.

Limitations

The analysis performed is restricted by the sample size, since there are only 5 years of data available. This methodology puts in practice only basic methods, since it is a preliminary approach. Thus, there are multiple further analysis that could be conducted. To the purpose of this approach, the levels of accuracy accepted were not too demanding. In terms of practical results, there might be some flaws since the sales were only considered in volume and not in value. Furthermore, in all the cases that required differencing, some real characteristics of the data might have been lost. Moreover, while building the Python code, simplifications to the statistical methods were made as otherwise no automation to support the application of the methodology would be feasible. Despite some loss of quality control on statistical methods,

automation versus the alternative of doing it analytically, was essential for practical application of the methodology.

Conclusions

The goal of this thesis is to provide decision makers with a methodology that supports strategic decisions to influence product sales. Variables that have statistical significance to explain the sales of each product were identified through the analysis.

The work project begins with an explanation of the importance that these models have to a consumer oriented company and a brief description of Beiersdorf. Then, it carries on to a definition of the external variables that can have an impact on product sales. Also, a short description about the three model types used is given. Next, the developed step-by-step methodology is presented, followed by a summary of the results of its application to all the 100 products of client 2. From all these products, the methodology failed to give accurate results to only 6 of them. Thereupon, four examples of product sales modeling are given. Firstly, sales of *product 24* were defined by a multiple regression. Then both products 75 and 76 sales were modeled respectively through an univariate ARIMA(0,1,1) and ARIMA(1,0,1). Finally, *product 90* time series are clearly seasonal so it was immediatly decided to follow a dynamic model, being the best-fit a SARIMA(0, 0, 0)x(2, 0, 0, 12) with one exogenous variable.

This work project provides tools to accurately explain the behavior of consumer product sales by applying these models to different types of products using Beiersdorf's data. Being a preliminary approach, some flwas were identified which may be improved with further investigation.

References

Hyndman, and George Athanasopoulos. 2014. *Forecasting: principles and practice*. OTexts.

McKinney, Wes. 2018. *Python for data analysis : data wrangling with pandas, NumPy, and IPython*. Boston: O'Reilly Media, Inc.

Beiersdorf. 2019. "Financial Reports". Accessed September 20. https://www.beiersdorf.com/investors/financial-reports/financial-reports (https://www.beiersdorf.com/investors/financial-reports/financial-reports)

Essays, UK. 2018. "The Marketing Planing Process Marketing Essay". Acessed September 20. https://www.ukessays.com/essays/marketing/the-marketing-planing-process-marketing-essay.php?vref=1 (https://www.ukessays.com/essays/marketing/the-marketing-planing-process-marketing-essay.php?vref=1)

Pordata 2018. “Consumption by households in the economic territory as % of GDP”. Retrived from pordata database https://www.pordata.pt/en/ (https://www.pordata.pt/en/)

Pordata 2018. “Real GDP growth rate”. Retrived from pordata database https://www.pordata.pt/en/ (https://www.pordata.pt/en/)

Pordata 2018. “Unemployment: total and by sex”. Retrived from pordata database https://www.pordata.pt/en/ (https://www.pordata.pt/en/)

Pordata 2018. “Guests in tourist accommodations per 100 inhabitants”. Retrived from pordata database https://www.pordata.pt/en/ (https://www.pordata.pt/en/)

Vincent, T. 2017. "ARIMA Time Series Data Forecasting and Visualization in Python". DigitalOcean, 2017. Retrieved 2 October 2019, from https://www.digitalocean.com/community/tutorials/a-guide-to-time-series-forecasting-with-arima-in-python-3 (https://www.digitalocean.com/community/tutorials/a-guide-to-time-series-forecasting-with-arima-in-python-3)

Machine Learning Plus. 2019. "ARIMA Model - Complete Guide to Time Series Forecasting in Python". Retrieved 7 October 2019, from https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/ (https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/)

Abbott, M.G. 2019. ECON 351* -- Note 2: OLS Estimation of the Simple CLRM, PDF. Retrieved from http://qed.econ.queensu.ca/pub/faculty/abbott/econ351/351note02.pdf (http://qed.econ.queensu.ca/pub/faculty/abbott/econ351/351note02.pdf)

ClockBackward. 2009. "Ordinary Least Squares Linear Regression: Flaws, Problems and Pitfalls". ClockBackward Essays.

Weisang and Awazu. 2008. "Vagaries of the Euro: an Introduction to ARIMA Modeling". Bentley College, 2008. CS-BIGS 2(1): 45-55. http://www.bentley.edu/csbigs/vol2-1/jaggia.pdf (http://www.bentley.edu/csbigs/vol2-1/jaggia.pdf)

Kilimci, Z., Akyuz, A., Uysal, M., Akyokus, S., Uysal, M., Atak Bulbul, B., & Ekmis, M. 2019. "An Improved Demand Forecasting Model Using Deep Learning Approach and Proposed Decision Integration Strategy for Supply Chain". Complexity, 2019, 1-15. doi: 10.1155/2019/9067367

Statsmodels v0.10.2 Documentation. 2019. Retrieved October 4. http://www.statsmodels.org (http://www.statsmodels.org)

Statwing Documentation. 2019. "Interpreting residual plots to improve your regression". Accessed October 4. http://docs.statwing.com/interpreting-residual-plots-to-improve-your-regression/ (http://docs.statwing.com/interpreting-residual-plots-to-improve-your-regression/)

Larrosa, Raphael Bubolz. 2019. "How to forecast sales with Python using SARIMA model. A step-by-step guide of statistic and python to time series forecasting". Towards Data Science, 2019. https://towardsdatascience.com/how-to-forecast-sales-with-python-using-sarima-model-

ba600992fa7d (https://towardsdatascience.com/how-to-forecast-sales-with-python-using-sarima-model-ba600992fa7d)

Sangarshanan. 2018. "Time series Forecasting — ARIMA models". Towards Data Science, 2018. https://towardsdatascience.com/time-series-forecasting-arima-models-7f221e9eee06 (https://towardsdatascience.com/time-series-forecasting-arima-models-7f221e9eee06)

Hatalis Kostas, 2018. "Tutorial: Multistep Forecasting with Seasonal ARIMA in Python". Data Science Central, 2018. https://www.datasciencecentral.com/profiles/blogs/tutorial-forecasting-with-seasonal-arima (https://www.datasciencecentral.com/profiles/blogs/tutorial-forecasting-with-seasonal-arima)

Nau, Robert. 2019. "What's a good value for R-squared?". Statistical forecasting: notes on regression and time series analysis, 2019. https://people.duke.edu/~rnau/rsquared.htm (https://people.duke.edu/~rnau/rsquared.htm)

PyFlux 0.4.7 documentation. 2019. "ARIMAX models" Accessed October 10. https://pyflux.readthedocs.io/en/latest/arimax.html (https://pyflux.readthedocs.io/en/latest/arimax.html)

Andy, 2019. "Logarithmic Transformation in Linear Regression Models: Why & When". DEV, 2019. Accessed October 4. https://dev.to/rokaandy/logarithmic-transformation-in-linear-regression-models-why-when-3a7c (https://dev.to/rokaandy/logarithmic-transformation-in-linear-regression-models-why-when-3a7c)

Venkat. 2019. "Time series forecasting with ARIMA using two different data sets". Bizstats.ai, 2019. https://www.bizstats.ai/blog/2019/02/20/time-series-forecasting-with-arima-using-two-different-data-sets/ (https://www.bizstats.ai/blog/2019/02/20/time-series-forecasting-with-arima-using-two-different-data-sets/)

Abugaber, David. 2019. "Chapter 23: Using ARIMA for Time Series Analysis". A language, not a letter: Learning statistics in R, 2019. https://ademos.people.uic.edu/index.html (https://ademos.people.uic.edu/index.html)

Johansen, Riani, Anthony C. Atkinson. 2012. "The Selection of ARIMA Models with or without Regressors". Department of Statistics, London School of Economics, 2012. https://econ.au.dk/fileadmin/site_files/filer_oekonomi/Working_Papers/CREATES/2012/rp12_46 (https://econ.au.dk/fileadmin/site_files/filer_oekonomi/Working_Papers/CREATES/2012/rp12_4

Stackoverflow. 2019. Retrieved from https://stackoverflow.com/ (https://stackoverflow.com/)

GitHub. 2019. https://github.github.com/ (https://github.github.com/)

Appendix

A. Overall results

Table A.1: The most acurate model for each product for client 2.

	Product	Method	Model detail
1	Dynamic	SARIMAX(1, 1, 0)x(1, 0, 0, 12)	
2	Dynamic	SARIMAX(0, 0, 0)x(2, 0, 1, 12)	
3	Dynamic	SARIMAX(1, 0, 0)x(1, 0, 1, 12)	
4	Dynamic	ARIMAX(1, 0, 2)	
5	Regression	Log-Lin	
6	Dynamic	ARIMAX(0, 0, 1)	
7	SARIMA Univariate	SARIMAX(0, 1, 1)x(1, 0, 0, 12)	
8	Regression	Lin-Lin	
9	Regression	Lin-Lin	
10	SARIMA Univariate	SARIMAX(0, 0, 0)x(2, 0, 0, 12)	
11	Dynamic	SARIMAX(0, 0, 0)x(0, 0, 1, 12)	
12	Regression	Lin-Lin	
13	Dynamic	ARIMAX(0, 0, 1)	
14	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 0, 12)	
15	ARIMA Univariate	ARIMA(0, 1, 1)	
16	ARIMA Univariate	ARIMA(0, 0, 1)	
17	None		
18	Dynamic	ARIMA(0, 1, 1)	
19	Regression	Log-Lin	
20	Regression	Log-Log	
21	Regression	Log-Lin	
22	Regression	Log-Lin	
23	Regression	Log-Log	
24	Regression	Log-Lin	
25	SARIMA Univariate	SARIMAX(0, 0, 0)x(0, 0, 1, 12)	
26	Regression	Lin-Log	
27	Dynamic	ARIMA(1, 0, 0)	
28	Regression	Log-Log	

	Product	Method	Model detail
29	SARIMA Univariate	SARIMAX(0, 0, 0)x(1, 0, 0, 12)	
30	None		
31	None		
32	Dynamic	ARIMA(0, 0, 1)	
33	Dynamic	ARIMA(0, 0, 1)	
34	ARIMA Univariate	ARIMA(1, 0, 1)	
35	None		
36	Regression	Log-Log	
37	Regression	Lin-Log	
38	None		
39	Dynamic	ARIMA(1, 1, 1)	
40	SARIMA Univariate	SARIMAX(0, 0, 0)x(1, 0, 0, 8)	
41	Regression	Lin-Lin	
42	Dynamic	ARIMA(0, 1, 1)	
43	ARIMA Univariate	ARIMA(0, 0, 1)	
44	ARIMA Univariate	ARIMA(1, 1, 1)	
45	Regression	Lin-Log	
46	Regression	Log-Lin	
47	Regression	Lin-Lin	
48	Regression	Log-Lin	
49	Regression	Log-Log	
50	Regression	Log-Log	
51	SARIMA Univariate	SARIMAX(0, 0, 0)x(0, 0, 1, 12)	
52	Dynamic	SARIMAX(0, 0, 0)x(0, 0, 1, 12)	
53	None		
54	Regression	Log-Lin	
55	Dynamic	ARIMA(0,0,2)	
56	Regression	Lin-Log	
57	Regression	Log-Log	
58	Regression	Log-Lin	
59	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 0, 12)	
60	Dynamic	ARIMA(0,0,1)	
61	Regression	Log-Lin	

	Product	Method	Model detail
62	Dynamic	ARIMAX(1, 0, 1)	
63	Dynamic	ARIMAX(0, 1, 2)	
64	SARIMA Univariate	SARIMAX(0, 0, 0)x(1, 0, 0, 12)	
65	Dynamic	ARIMAX(0, 0, 2)	
66	Dynamic	SARIMAX(0, 1, 1)x(1, 0, 0, 8)	
67	Dynamic	SARIMAX(0, 0, 0)x(3, 0, 1, 6)	
68	Dynamic	ARIMAX(0, 1, 1)	
69	Dynamic	ARIMAX(1, 0, 1)	
70	Dynamic	ARIMAX(1, 0, 1)	
71	ARIMA Univariate	ARIMA(0, 1, 1)	
72	ARIMA Univariate	ARIMA(1, 1, 1)	
73	SARIMA Univariate	SARIMAX(0, 0, 1)x(0, 0, 1, 12)	
74	Regression	Log-Log	
75	Dynamic	ARIMA(0, 1, 1)	
76	ARIMA Univariate	ARIMA(1, 0, 1)	
77	ARIMA Univariate	ARIMA(1, 1, 2)	
78	Dynamic	ARIMA(0, 1, 1)	
79	ARIMA Univariate	ARIMA(0, 1, 1)	
80	ARIMA Univariate	ARIMA(1, 1, 1)	
81	Dynamic	ARIMA(0, 1, 1)	
82	Regression	Lin-Log	
83	ARIMA Univariate	ARIMA(0, 1, 1)	
84	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 1, 12)	
85	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 0, 12)	
86	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 0, 12)	
87	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 0, 12)	
88	Dynamic	SARIMAX(0, 0, 0)x(2, 0, 0, 12)	
89	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 1, 12)	
90	Dynamic	SARIMAX(0, 0, 0)x(2, 0, 0, 12)	
91	Dynamic	SARIMAX(0, 0, 0)x(2, 0, 0, 12)	
92	Dynamic	SARIMAX(1, 0, 1)x(2, 0, 0, 12)	
93	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 1, 12)	
94	Dynamic	SARIMAX(1, 0, 1)x(2, 0, 0, 12)	

	Product	Method	Model detail
95	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 0, 12)	
96	Dynamic	SARIMAX(1, 0, 0)x(1, 0, 1, 12)	
97	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 0, 12)	
98	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 1, 12)	
99	Dynamic	SARIMAX(0, 0, 0)x(2, 0, 0, 12)	
100	Dynamic	SARIMAX(0, 0, 0)x(1, 0, 0, 12)	

B. Experimental results - Multiple Regression

Figure B.1: Best-fit model from transormations and forward selecting strategy

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-11.7556	97.994	-0.120	0.905	-208.486	184.975
e2	0.1596	0.218	0.733	0.467	-0.277	0.596
MP253	14.1155	49.598	0.285	0.777	-85.457	113.688
C31	0.3057	0.332	0.920	0.362	-0.361	0.973
MP251	36.9554	30.544	1.210	0.232	-24.363	98.274
MS23	9.5178	3.853	2.470	0.017	1.783	17.253
MP242	182.6532	40.277	4.535	0.000	101.795	263.512
MP200	18.4715	7.560	2.443	0.018	3.294	33.649
C11	-0.9454	0.487	-1.943	0.058	-1.922	0.032
MP263	-188.8325	53.431	-3.534	0.001	-296.099	-81.566
MP243	477.0924	107.796	4.426	0.000	260.682	693.502
MP261	143.9353	40.803	3.528	0.001	62.019	225.852
e1	-0.0556	1.333	-0.042	0.967	-2.732	2.621
C22	0.0945	0.031	3.016	0.004	0.032	0.157
PI22	2.2582	1.000	2.259	0.028	0.251	4.265
C13	-0.0474	0.027	-1.767	0.083	-0.101	0.006
C33	-0.0535	0.033	-1.641	0.107	-0.119	0.012
MS22	-16.2809	15.279	-1.066	0.292	-46.955	14.393

Dep. Variable:	logy	R-squared:	0.658
Model:	OLS	Adj. R-squared:	0.543
Method:	Least Squares	F-statistic:	5.761
Date:	Sun, 29 Dec 2019	Prob (F-statistic):	5.35e-07
Time:	14:21:02	Log-Likelihood:	-24.382
No. Observations:	69	AIC:	84.76
Df Residuals:	51	BIC:	125.0
Df Model:	17		
Covariance Type:	nonrobust		

Omnibus:	8.209	Durbin-Watson:	2.432
Prob(Omnibus):	0.016	Jarque-Bera (JB):	13.294
Skew:	0.329	Prob(JB):	0.00130
Kurtosis:	5.047	Cond. No.	1.02e+06

Figure B.2: Final best-fit model

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-8.5515	13.393	-0.638	0.526	-35.361	18.258
MS23	8.8471	2.688	3.291	0.002	3.467	14.227
MP232	209.7280	29.187	7.186	0.000	151.303	268.153
MP200	26.2855	5.752	4.570	0.000	14.773	37.798
C11	-0.7727	0.384	-2.013	0.049	-1.541	-0.004
MP253	-177.8003	48.713	-3.650	0.001	-275.310	-80.291
MP233	378.3133	88.089	4.295	0.000	201.984	554.642
MP251	149.6778	28.456	5.260	0.000	92.718	206.638
C22	0.0654	0.014	4.806	0.000	0.038	0.093
C13	-0.0286	0.011	-2.691	0.009	-0.050	-0.007
C33	-0.0524	0.020	-2.624	0.011	-0.092	-0.012

Dep. Variable:	logy	R-squared:	0.602
Model:	OLS	Adj. R-squared:	0.533
Method:	Least Squares	F-statistic:	8.772
Date:	Sun, 29 Dec 2019	Prob (F-statistic):	1.47e-08
Time:	14:47:51	Log-Likelihood:	-29.573
No. Observations:	69	AIC:	81.15
Df Residuals:	58	BIC:	105.7
Df Model:	10		
Covariance Type:	nonrobust		

Omnibus:	4.915	Durbin-Watson:	2.434
Prob(Omnibus):	0.086	Jarque-Bera (JB):	6.356
Skew:	-0.109	Prob(JB):	0.0417
Kurtosis:	4.471	Cond. No.	8.23e+05

C. Experimental results - ARIMA Univariate stationary

Figure C.1: Final best-fit model

Dep. Variable:	y	No. Observations:	69
Model:	ARMA(1, 0)	Log Likelihood	-399.116
Method:	css-mle	S.D. of innovations	78.572
Date:	Sun, 29 Dec 2019	AIC	804.231
Time:	17:47:59	BIC	810.934
Sample:	0	HQIC	806.891

	coef	std err	z	P>|z|	[0.025	0.975]
const	135.5775	15.878	8.539	0.000	104.457	166.698
ar.L1.y	0.4100	0.111	3.680	0.000	0.192	0.628

D. Experimental results - ARIMA Univariate non-stationary

Figure D.1: Final best-fit model

ARIMA Model Results

Dep. Variable:	D.y	**No. Observations:**	68
Model:	ARIMA(0, 1, 1)	**Log Likelihood**	-477.433
Method:	css-mle	**S.D. of innovations**	262.697
Date:	Sun, 29 Dec 2019	**AIC**	962.866
Time:	18:05:49	**BIC**	971.744
Sample:	1	**HQIC**	966.384

	coef	std err	z	P>|z|	[0.025	0.975]
const	37.5523	5.424	6.924	0.000	26.922	48.183
x1	-181.4459	33.016	-5.496	0.000	-246.155	-116.736
ma.L1.D.y	-1.0000	0.042	-23.540	0.000	-1.083	-0.917

Roots

	Real	Imaginary	Modulus	Frequency
MA.1	1.0000	+0.0000j	1.0000	0.0000

E. Experimental results - Dynamic model

Figure E.1: Final best-fit model

Dep. Variable:	y	**No. Observations:**	69
Model:	SARIMAX(2, 0, 0, 12)	**Log Likelihood**	-583.553
Date:	Sun, 29 Dec 2019	**AIC**	1177.105
Time:	18:31:46	**BIC**	1188.276
Sample:	0	**HQIC**	1181.537
	- 69		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	3.4704	100.207	0.035	0.972	-192.931	199.872
x1	2.951e+05	1.59e+05	1.856	0.063	-1.65e+04	6.07e+05
ar.S.L12	0.4767	0.090	5.269	0.000	0.299	0.654
ar.S.L24	0.2849	0.106	2.689	0.007	0.077	0.493
sigma2	1.144e+06	1.57e+05	7.264	0.000	8.35e+05	1.45e+06

Ljung-Box (Q):	27.67	**Jarque-Bera (JB):**	15.15
Prob(Q):	0.93	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.01	**Skew:**	-0.64
Prob(H) (two-sided):	0.10	**Kurtosis:**	4.90

F. Pyhton code

Tabke F.1: Python code for the methodology

```
pip install pmdarima

# Data curation
import pandas as pd
import numpy as np
%matplotlib notebook
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf

sales_volume = pd.read_excel('data_clean.xlsx', header=[0,1,2,
3,4], index_col=[0, 1], sheetname='Sales Volume')
sales_volume = pd.DataFrame(sales_volume)
sales_volume = sales_volume.replace(np.NaN, 0)


def get_tokens(string):
    import re
    tokens = re.sub(r'[^\sa-zA-Z0-9]', '', string).lower().str
ip().split()
    return tokens

#Variables definition

y_raw = pd.read_excel('data_clean.xlsx', header=[0,1,2,3,4], i
ndex_col=[0, 1], sheetname='Sales Volume')
y_raw = y_raw.replace(np.NaN, 0)
y_raw = pd.DataFrame(y_raw)
```

```
x1 = pd.read_excel('data_clean.xlsx', skiprows = 17, header=[0
,1], index_col=[0, 1], sheetname='Internal Data-Promo Intensit
y')
x1 = pd.DataFrame(x1)

x2 = pd.read_excel('data_clean.xlsx', skiprows = 32, header=[0
,1,2], index_col=[0, 1], sheetname='Internal Data-MarketPenetr
ation')
x2 = pd.DataFrame(x2)

x3 = pd.read_excel('data_clean.xlsx', skiprows = 16, header=[0
,1], index_col=[0, 1], sheetname='Internal Data-Market Share')
x3 = pd.DataFrame(x3)

x4 = pd.read_excel('data_clean.xlsx', skiprows = 7, header=[0,
1], index_col=[0, 1], sheetname='Stores per Client')
x4 = pd.DataFrame(x4)
x4cleancols1 = pd.DataFrame('x4'+ x4.iloc[:,0:8].columns.get_l
evel_values(0) + x4.iloc[:,0:8].columns.get_level_values(1))
x4cleancols2 = get_tokens(str(x4cleancols1))
x4cleancols = [x for x in x4cleancols2 if x.startswith('x4')]
x4.columns = x4cleancols

x5 = pd.read_excel('data_clean.xlsx', skiprows = 2, header=[0,
1,2], index_col=[0, 1], sheetname='Internal Data-Media Plan')
x5 = x5.replace(np.NaN, 0)
x5 = pd.DataFrame(x5)

e_n = pd.read_excel('data_clean.xlsx', skiprows = 24, header=[
```

```
0,1], index_col=[0, 1], sheetname='External Data')
e_n = pd.DataFrame(e_n)
e_n.columns = ['e1', 'e2', 'e3', 'e41', 'e42', 'e43', 'e5']

# Categories

#Y - sales time series:
aftersun_pd = y_raw.iloc[:, 0:4]
labello_pd = y_raw.iloc[:, 36:56]
bath_pd = y_raw.iloc[:, 57:68]
shower_pd = y_raw.iloc[:, 69:100]
inshower_pd = y_raw.iloc[:, 146:153]
soap_pd = y_raw.iloc[:,101:116]
bodyessentials_pd = y_raw.iloc[:, 117:146]
bodyperformance_pd = y_raw.iloc[:,154:166]
bodyapc_pd = y_raw.iloc[:,167:181]
facecare_pd = y_raw.iloc[:, 182:244]
facecleansing_pd = y_raw.iloc[:,245:289]
hand_pd = y_raw.iloc[:,290:302]
deofemale_pd = y_raw.iloc[:, 302:339]
deomale_pd = y_raw.iloc[:,340:367]
haircare_pd = y_raw.iloc[:, 368:401]
styling_pd = y_raw.iloc[:, 402:419]
aftershave_pd = y_raw.iloc[:, 420:431]
menfacecare_pd = y_raw.iloc[:, 432:465]
sunprotect_pd = y_raw.iloc[:,466:514]

aftersun_sonae = y_raw.iloc[:, 529:532]
```

```
x1facecleansing = x1.iloc[:, 4:8]
x1facecleansingcleancols1 =  pd.DataFrame('x1facecleansing'+ x
1facecleansing.columns.get_level_values(1))
x1facecleansingcleancols = get_tokens(str(x1facecleansingclean
cols1))
x1facecleansingcleancols = [x for x in x1facecleansingcleancol
s if x.startswith('x1')]
logx1facecleansing = np.log(x1.iloc[:, 4:8].replace(0,1))
logx1facecleansingcleancols1 =  pd.DataFrame('logx1facecleansi
ng'+ x1facecleansing.columns.get_level_values(1))
logx1facecleansingcleancols = get_tokens(str(logx1facecleansin
gcleancols1))
logx1facecleansingcleancols = [x for x in logx1facecleansingcl
eancols if x.startswith('logx1')]
logx1facecleansing.columns =logx1facecleansingcleancols
x1facecleansing.columns = x1facecleansingcleancols

x2face = x2.iloc[:, 0:10]
x2facecleancols1 =  pd.DataFrame('x2face'+ x2face.columns.get_
level_values(1)+ x2face.columns.get_level_values(2))
x2facecleancols = get_tokens(str(x2facecleancols1))
x2facecleancols = [x for x in x2facecleancols if x.startswith(
'x2')]
logx2face = np.log(x2.iloc[:, 0:10].replace(0,1))
logx2facecleancols1 =  pd.DataFrame('logx2face'+ x2face.column
s.get_level_values(1)+ x2face.columns.get_level_values(2))
logx2facecleancols = get_tokens(str(logx2facecleancols1))
logx2facecleancols = [x for x in logx2facecleancols if x.start
swith('logx2')]
```

```
logx2face.columns =logx2facecleancols
x2face.columns = x2facecleancols

x3facecleansing = x3.iloc[:, 3:6]
x3facecleansingcleancols1 =  pd.DataFrame('x3facecleansing'+ x
3facecleansing.columns.get_level_values(1))
x3facecleansingcleancols = get_tokens(str(x3facecleansingclean
cols1))
x3facecleansingcleancols = [x for x in x3facecleansingcleancol
s if x.startswith('x3')]
logx3facecleansing = np.log(x3.iloc[:, 3:6].replace(0,1))
logx3facecleansingcleancols1 =  pd.DataFrame('logx3facecleansi
ng'+ x3facecleansing.columns.get_level_values(1))
logx3facecleansingcleancols = get_tokens(str(logx3facecleansin
gcleancols1))
logx3facecleansingcleancols = [x for x in logx3facecleansingcl
eancols if x.startswith('logx3')]
logx3facecleansing.columns =logx3facecleansingcleancols
x3facecleansing.columns = x3facecleansingcleancols

x1facecare = x1.iloc[:, 9:12]
x1facecarecleancols1 =  pd.DataFrame('x1facecare'+ x1facecare.
columns.get_level_values(1))
x1facecarecleancols = get_tokens(str(x1facecarecleancols1))
x1facecarecleancols = [x for x in x1facecarecleancols if x.sta
rtswith('x1')]
logx1facecare = np.log(x1.iloc[:, 9:12].replace(0,1))
logx1facecarecleancols1 =  pd.DataFrame('logx1facecare'+ x1fac
ecare.columns.get_level_values(1))
```

```
logx1facecarecleancols = get_tokens(str(logx1facecarecleancols
1))
logx1facecarecleancols = [x for x in logx1facecarecleancols if
x.startswith('logx1')]
logx1facecare.columns =logx1facecarecleancols
x1facecare.columns = x1facecarecleancols

x3facecare = x3.iloc[:, 6:9]
x3facecarecleancols1 =  pd.DataFrame('x3facecare'+ x3facecare.
columns.get_level_values(1))
x3facecarecleancols = get_tokens(str(x3facecarecleancols1))
x3facecarecleancols = [x for x in x3facecarecleancols if x.sta
rtswith('x3')]
logx3facecare = np.log(x3.iloc[:, 6:9].replace(0,1))
logx3facecarecleancols1 =  pd.DataFrame('logx3facecare'+ x3fac
ecare.columns.get_level_values(1))
logx3facecarecleancols = get_tokens(str(logx3facecarecleancols
1))
logx3facecarecleancols = [x for x in logx3facecarecleancols if
x.startswith('logx3')]
logx3facecare.columns =logx3facecarecleancols
x3facecare.columns = x3facecarecleancols

x1bodyessentials = x1.iloc[:, 16:20]
x1bodyessentialscleancols1 =  pd.DataFrame('x1bodyessentials'+
x1bodyessentials.columns.get_level_values(1))
x1bodyessentialscleancols = get_tokens(str(x1bodyessentialscle
ancols1))
x1bodyessentialscleancols = [x for x in x1bodyessentialscleanc
```

```
ols if x.startswith('x1')]
logx1bodyessentials = np.log(x1.iloc[:, 16:20].replace(0,1))
logx1bodyessentialscleancols1 =  pd.DataFrame('logx1bodyessent
ials'+ x1bodyessentials.columns.get_level_values(1))
logx1bodyessentialscleancols = get_tokens(str(logx1bodyessenti
alscleancols1))
logx1bodyessentialscleancols = [x for x in logx1bodyessentials
cleancols if x.startswith('logx1')]
logx1bodyessentials.columns =logx1bodyessentialscleancols
x1bodyessentials.columns = x1bodyessentialscleancols

x2body = x2.iloc[:, 10:26]
x2bodycleancols1 =  pd.DataFrame('x2body'+ x2body.columns.get_
level_values(1)+ x2body.columns.get_level_values(2))
x2bodycleancols = get_tokens(str(x2bodycleancols1))
x2bodycleancols = [x for x in x2bodycleancols if x.startswith(
'x2')]
logx2body = np.log(x2.iloc[:, 10:26].replace(0,1))
logx2bodycleancols1 =  pd.DataFrame('logx2body'+ x2body.column
s.get_level_values(1)+ x2body.columns.get_level_values(2))
logx2bodycleancols = get_tokens(str(logx2bodycleancols1))
logx2bodycleancols = [x for x in logx2bodycleancols if x.start
swith('logx2')]
logx2body.columns =logx2bodycleancols
x2body.columns = x2bodycleancols

x3bodyessentials = x3.iloc[:, 12:15]
x3bodyessentialscleancols1 =  pd.DataFrame('x3bodyessentials'+
x3bodyessentials.columns.get_level_values(1))
```

```
x3bodyessentialscleancols = get_tokens(str(x3bodyessentialscle
ancols1))
x3bodyessentialscleancols = [x for x in x3bodyessentialscleanc
ols if x.startswith('x3')]
logx3bodyessentials = np.log(x3.iloc[:, 12:15].replace(0,1))
logx3bodyessentialscleancols1 =  pd.DataFrame('logx3bodyessent
ials'+ x3bodyessentials.columns.get_level_values(1))
logx3bodyessentialscleancols = get_tokens(str(logx3bodyessenti
alscleancols1))
logx3bodyessentialscleancols = [x for x in logx3bodyessentials
cleancols if x.startswith('logx3')]
logx3bodyessentials.columns =logx3bodyessentialscleancols
x3bodyessentials.columns = x3bodyessentialscleancols

x1bodyperformance = x1.iloc[:, 20:24]
x1bodyperformancecleancols1 =  pd.DataFrame('x1bodyperformanc
e'+ x1bodyperformance.columns.get_level_values(1))
x1bodyperformancecleancols = get_tokens(str(x1bodyperformancec
leancols1))
x1bodyperformancecleancols = [x for x in x1bodyperformanceclea
ncols if x.startswith('x1')]
logx1bodyperformance = np.log(x1.iloc[:, 20:24].replace(0,1))
logx1bodyperformancecleancols1 =  pd.DataFrame('logx1bodyperfo
rmance'+ x1bodyperformance.columns.get_level_values(1))
logx1bodyperformancecleancols = get_tokens(str(logx1bodyperfor
mancecleancols1))
logx1bodyperformancecleancols = [x for x in logx1bodyperforman
cecleancols if x.startswith('logx1')]
logx1bodyperformance.columns =logx1bodyperformancecleancols
```

```
x1bodyperformance.columns = x1bodyperformancecleancols

x3bodyperformance = x3.iloc[:, 15:18]
x3bodyperformancecleancols1 =   pd.DataFrame('x3bodyperformanc
e'+ x3bodyperformance.columns.get_level_values(1))
x3bodyperformancecleancols = get_tokens(str(x3bodyperformancec
leancols1))
x3bodyperformancecleancols = [x for x in x3bodyperformanceclea
ncols if x.startswith('x3')]
logx3bodyperformance = np.log(x3.iloc[:, 15:18].replace(0,1))
logx3bodyperformancecleancols1 =  pd.DataFrame('logx3bodyperfo
rmance'+ x3bodyperformance.columns.get_level_values(1))
logx3bodyperformancecleancols = get_tokens(str(logx3bodyperfor
mancecleancols1))
logx3bodyperformancecleancols = [x for x in logx3bodyperforman
cecleancols if x.startswith('logx3')]
logx3bodyperformance.columns =logx3bodyperformancecleancols
x3bodyperformance.columns = x3bodyperformancecleancols

x1bodyapc = x1.iloc[:, 24:28]
x1bodyapccleancols1 =  pd.DataFrame('x1bodyapc'+ x1bodyapc.col
umns.get_level_values(1))
x1bodyapccleancols = get_tokens(str(x1bodyapccleancols1))
x1bodyapccleancols = [x for x in x1bodyapccleancols if x.start
swith('x1')]
logx1bodyapc = np.log(x1.iloc[:, 24:28].replace(0,1))
logx1bodyapccleancols1 =  pd.DataFrame('logx1bodyapc'+ x1bodya
pc.columns.get_level_values(1))
logx1bodyapccleancols  =  get_tokens(str(logx1bodyapccleancols1
```

```
))
logx1bodyapccleancols = [x for x in logx1bodyapccleancols if x
.startswith('logx1')]
logx1bodyapc.columns =logx1bodyapccleancols
x1bodyapc.columns = x1bodyapccleancols

x3bodyapc = x3.iloc[:, 18:21]
x3bodyapccleancols1 =  pd.DataFrame('x3bodyapc'+ x3bodyapc.col
umns.get_level_values(1))
x3bodyapccleancols = get_tokens(str(x3bodyapccleancols1))
x3bodyapccleancols = [x for x in x3bodyapccleancols if x.start
swith('x3')]
logx3bodyapc = np.log(x3.iloc[:, 18:21].replace(0,1))
logx3bodyapccleancols1 =  pd.DataFrame('logx3bodyapc'+ x3bodya
pc.columns.get_level_values(1))
logx3bodyapccleancols  =  get_tokens(str(logx3bodyapccleancols1
))
logx3bodyapccleancols = [x for x in logx3bodyapccleancols if x
.startswith('logx3')]
logx3bodyapc.columns =logx3bodyapccleancols
x3bodyapc.columns = x3bodyapccleancols

x1shower = x1.iloc[:, 28:32]
x1showercleancols1 =  pd.DataFrame('x1shower'+ x1shower.column
s.get_level_values(1))
x1showercleancols = get_tokens(str(x1showercleancols1))
x1showercleancols = [x for x in x1showercleancols if x.startsw
ith('x1')]
logx1shower = np.log(x1.iloc[:, 28:32].replace(0,1))
```

```
logx1showercleancols1 =  pd.DataFrame('logx1shower'+ x1shower.
columns.get_level_values(1))
logx1showercleancols = get_tokens(str(logx1showercleancols1))
logx1showercleancols = [x for x in logx1showercleancols if x.s
tartswith('logx1')]
logx1shower.columns =logx1showercleancols
x1shower.columns = x1showercleancols

x2shower = x2.iloc[:, 45:58]
x2showercleancols1 =  pd.DataFrame('x2shower'+ x2shower.column
s.get_level_values(1)+ x2shower.columns.get_level_values(2))
x2showercleancols = get_tokens(str(x2showercleancols1))
x2showercleancols = [x for x in x2showercleancols if x.startsw
ith('x2')]
logx2shower = np.log(x2.iloc[:, 45:58].replace(0,1))
logx2showercleancols1 =  pd.DataFrame('logx2shower'+ x2shower.
columns.get_level_values(1)+  x2shower.columns.get_level_values
(2))
logx2showercleancols = get_tokens(str(logx2showercleancols1))
logx2showercleancols = [x for x in logx2showercleancols if x.s
tartswith('logx2')]
logx2shower.columns =logx2showercleancols
x2shower.columns = x2showercleancols

x3shower = x3.iloc[:, 21:24]
x3showercleancols1 =  pd.DataFrame('x3shower'+ x3shower.column
s.get_level_values(1))
x3showercleancols = get_tokens(str(x3showercleancols1))
x3showercleancols = [x for x in x3showercleancols if x.startsw
```

```
ith('x3')]
logx3shower = np.log(x3.iloc[:, 21:24].replace(0,1))
logx3showercleancols1 =  pd.DataFrame('logx3shower'+ x3shower.
columns.get_level_values(1))
logx3showercleancols = get_tokens(str(logx3showercleancols1))
logx3showercleancols = [x for x in logx3showercleancols if x.s
tartswith('logx3')]
logx3shower.columns =logx3showercleancols
x3shower.columns = x3showercleancols

x1deofemale = x1.iloc[:, 37:41]
x1deofemalecleancols1 =  pd.DataFrame('x1deofemale'+ x1deofema
le.columns.get_level_values(1))
x1deofemalecleancols = get_tokens(str(x1deofemalecleancols1))
x1deofemalecleancols = [x for x in x1deofemalecleancols if x.s
tartswith('x1')]
logx1deofemale = np.log(x1.iloc[:, 37:41].replace(0,1))
logx1deofemalecleancols1 =  pd.DataFrame('logx1deofemale'+ x1d
eofemale.columns.get_level_values(1))
logx1deofemalecleancols = get_tokens(str(logx1deofemalecleanco
ls1))
logx1deofemalecleancols = [x for x in logx1deofemalecleancols
if x.startswith('logx1')]
logx1deofemale.columns =logx1deofemalecleancols
x1deofemale.columns = x1deofemalecleancols

x2deo = x2.iloc[:, 27:46]
x2deocleancols1 =  pd.DataFrame('x2deo'+ x2deo.columns.get_lev
el_values(1)+ x2deo.columns.get_level_values(2))
```

```
x2deocleancols = get_tokens(str(x2deocleancols1))
x2deocleancols = [x for x in x2deocleancols if x.startswith('x
2')]
logx2deo = np.log(x2.iloc[:, 27:46].replace(0,1))
logx2deocleancols1 =  pd.DataFrame('logx2deo'+ x2deo.columns.g
et_level_values(1)+ x2deo.columns.get_level_values(2))
logx2deocleancols = get_tokens(str(logx2deocleancols1))
logx2deocleancols = [x for x in logx2deocleancols if x.startsw
ith('logx2')]
logx2deo.columns =logx2deocleancols
x2deo.columns = x2deocleancols

x3deofemale = x3.iloc[:, 28:31]
x3deofemalecleancols1 =  pd.DataFrame('x3deofemale'+ x3deofema
le.columns.get_level_values(1))
x3deofemalecleancols = get_tokens(str(x3deofemalecleancols1))
x3deofemalecleancols = [x for x in x3deofemalecleancols if x.s
tartswith('x3')]
logx3deofemale = np.log(x3.iloc[:, 28:31].replace(0,1))
logx3deofemalecleancols1 =  pd.DataFrame('logx3deofemale'+ x3d
eofemale.columns.get_level_values(1))
logx3deofemalecleancols = get_tokens(str(logx3deofemalecleanco
ls1))
logx3deofemalecleancols = [x for x in logx3deofemalecleancols
if x.startswith('logx3')]
logx3deofemale.columns =logx3deofemalecleancols
x3deofemale.columns = x3deofemalecleancols

x1deomale = x1.iloc[:, 40:44]
```

```
x1deomalecleancols1 =  pd.DataFrame('x1deomale'+ x1deomale.col
umns.get_level_values(1))
x1deomalecleancols = get_tokens(str(x1deomalecleancols1))
x1deomalecleancols = [x for x in x1deomalecleancols if x.start
swith('x1')]
logx1deomale = np.log(x1.iloc[:, 40:44].replace(0,1))
logx1deomalecleancols1 =  pd.DataFrame('logx1deomale'+ x1deoma
le.columns.get_level_values(1))
logx1deomalecleancols  =  get_tokens(str(logx1deomalecleancols1
))
logx1deomalecleancols = [x for x in logx1deomalecleancols if x
.startswith('logx1')]
logx1deomale.columns =logx1deomalecleancols
x1deomale.columns = x1deomalecleancols

x3deomale = x3.iloc[:, 30:33]
x3deomalecleancols1 =  pd.DataFrame('x3deomale'+ x3deomale.col
umns.get_level_values(1))
x3deomalecleancols = get_tokens(str(x3deomalecleancols1))
x3deomalecleancols = [x for x in x3deomalecleancols if x.start
swith('x3')]
logx3deomale = np.log(x3.iloc[:, 30:33].replace(0,1))
logx3deomalecleancols1 =  pd.DataFrame('logx3deomale'+ x3deoma
le.columns.get_level_values(1))
logx3deomalecleancols  =  get_tokens(str(logx3deomalecleancols1
))
logx3deomalecleancols = [x for x in logx3deomalecleancols if x
.startswith('logx3')]
logx3deomale.columns =logx3deomalecleancols
```

```
x3deomale.columns = x3deomalecleancols


x1menaftershave = x1.iloc[:, 48:52]
x1menaftershavecleancols1 =  pd.DataFrame('x1menaftershave'+ x
1menaftershave.columns.get_level_values(1))
x1menaftershavecleancols = get_tokens(str(x1menaftershaveclean
cols1))
x1menaftershavecleancols = [x for x in x1menaftershavecleancol
s if x.startswith('x1')]
logx1menaftershave = np.log(x1.iloc[:, 48:52].replace(0,1))
logx1menaftershavecleancols1 =  pd.DataFrame('logx1menaftersha
ve'+ x1menaftershave.columns.get_level_values(1))
logx1menaftershavecleancols = get_tokens(str(logx1menaftershav
ecleancols1))
logx1menaftershavecleancols = [x for x in logx1menaftershavecl
eancols if x.startswith('logx1')]
logx1menaftershave.columns =logx1menaftershavecleancols
x1menaftershave.columns = x1menaftershavecleancols


x2menaftershave = x2.iloc[:, 87:103]
x2menaftershavecleancols1 =  pd.DataFrame('x2menaftershave'+ x
2menaftershave.columns.get_level_values(1)+  x2menaftershave.co
lumns.get_level_values(2))
x2menaftershavecleancols = get_tokens(str(x2menaftershaveclean
cols1))
x2menaftershavecleancols = [x for x in x2menaftershavecleancol
s if x.startswith('x2')]
logx2menaftershave = np.log(x2.iloc[:, 87:103].replace(0,1))
logx2menaftershavecleancols1 =  pd.DataFrame('logx2menaftersha
```

```
ve'+ x2menaftershave.columns.get_level_values(1)+ x2menaftersh
ave.columns.get_level_values(2))
logx2menaftershavecleancols = get_tokens(str(logx2menaftershav
ecleancols1))
logx2menaftershavecleancols = [x for x in logx2menaftershavecl
eancols if x.startswith('logx2')]
logx2menaftershave.columns =logx2menaftershavecleancols
x2menaftershave.columns = x2menaftershavecleancols

x3menaftershave = x3.iloc[:, 36:39]
x3menaftershavecleancols1 =  pd.DataFrame('x3menaftershave'+ x
3menaftershave.columns.get_level_values(1))
x3menaftershavecleancols = get_tokens(str(x3menaftershaveclean
cols1))
x3menaftershavecleancols = [x for x in x3menaftershavecleancol
s if x.startswith('x3')]
logx3menaftershave = np.log(x3.iloc[:, 36:39].replace(0,1))
logx3menaftershavecleancols1 =  pd.DataFrame('logx3menaftersha
ve'+ x3menaftershave.columns.get_level_values(1))
logx3menaftershavecleancols = get_tokens(str(logx3menaftershav
ecleancols1))
logx3menaftershavecleancols = [x for x in logx3menaftershavecl
eancols if x.startswith('logx3')]
logx3menaftershave.columns =logx3menaftershavecleancols
x3menaftershave.columns = x3menaftershavecleancols

x1menfacecare = x1.iloc[:, 52:56]
x1menfacecarecleancols1 =  pd.DataFrame('x1menfacecare'+ x1men
facecare.columns.get_level_values(1))
```

```
x1menfacecarecleancols = get_tokens(str(x1menfacecarecleancols
1))
x1menfacecarecleancols = [x for x in x1menfacecarecleancols if
x.startswith('x1')]
logx1menfacecare = np.log(x1.iloc[:, 52:56].replace(0,1))
logx1menfacecarecleancols1 =  pd.DataFrame('logx1menfacecare'+
x1menfacecare.columns.get_level_values(1))
logx1menfacecarecleancols = get_tokens(str(logx1menfacecarecle
ancols1))
logx1menfacecarecleancols = [x for x in logx1menfacecarecleanc
ols if x.startswith('logx1')]
logx1menfacecare.columns =logx1menfacecarecleancols
x1menfacecare.columns = x1menfacecarecleancols

x2menfacecare = x2.iloc[:, 103:113]
x2menfacecarecleancols1 =  pd.DataFrame('x2menfacecare'+ x2men
facecare.columns.get_level_values(1)+ x2menfacecare.columns.ge
t_level_values(2))
x2menfacecarecleancols = get_tokens(str(x2menfacecarecleancols
1))
x2menfacecarecleancols = [x for x in x2menfacecarecleancols if
x.startswith('x2')]
logx2menfacecare = np.log(x2.iloc[:, 103:113].replace(0,1))
logx2menfacecarecleancols1 =  pd.DataFrame('logx2menfacecare'+
x2menfacecare.columns.get_level_values(1)+ x2menfacecare.colum
ns.get_level_values(2))
logx2menfacecarecleancols = get_tokens(str(logx2menfacecarecle
ancols1))
logx2menfacecarecleancols = [x for x in logx2menfacecarecleanc
```

```
ols if x.startswith('logx2')]
logx2menfacecare.columns =logx2menfacecarecleancols
x2menfacecare.columns = x2menfacecarecleancols

x3menfacecare = x3.iloc[:, 39:42]
x3menfacecarecleancols1 =  pd.DataFrame('x3menfacecare'+ x3men
facecare.columns.get_level_values(1))
x3menfacecarecleancols = get_tokens(str(x3menfacecarecleancols
1))
x3menfacecarecleancols = [x for x in x3menfacecarecleancols if
x.startswith('x3')]
logx3menfacecare = np.log(x3.iloc[:, 39:42].replace(0,1))
logx3menfacecarecleancols1 =  pd.DataFrame('logx3menfacecare'+
x3menfacecare.columns.get_level_values(1))
logx3menfacecarecleancols = get_tokens(str(logx3menfacecarecle
ancols1))
logx3menfacecarecleancols = [x for x in logx3menfacecarecleanc
ols if x.startswith('logx3')]
logx3menfacecare.columns =logx3menfacecarecleancols
x3menfacecare.columns = x3menfacecarecleancols

x1sunprotection = x1.iloc[:, 60:64]
x1sunprotectioncleancols1 =  pd.DataFrame('x1sunprotection'+ x
1sunprotection.columns.get_level_values(1))
x1sunprotectioncleancols = get_tokens(str(x1sunprotectionclean
cols1))
x1sunprotectioncleancols = [x for x in x1sunprotectioncleancol
s if x.startswith('x1')]
logx1sunprotection = np.log(x1.iloc[:, 60:64].replace(0,1))
```

```
logx1sunprotectioncleancols1 =  pd.DataFrame('logx1sunprotecti
on'+ x1sunprotection.columns.get_level_values(1))
logx1sunprotectioncleancols = get_tokens(str(logx1sunprotectio
ncleancols1))
logx1sunprotectioncleancols = [x for x in logx1sunprotectioncl
eancols if x.startswith('logx1')]
logx1sunprotection.columns =logx1sunprotectioncleancols
x1sunprotection.columns = x1sunprotectioncleancols

x2sun = x2.iloc[:, 113:126]
x2suncleancols1 =  pd.DataFrame('x2sun'+ x2sun.columns.get_lev
el_values(1)+ x2sun.columns.get_level_values(2))
x2suncleancols = get_tokens(str(x2suncleancols1))
x2suncleancols = [x for x in x2suncleancols if x.startswith('x
2')]
logx2sun = np.log(x2.iloc[:, 113:126].replace(0,1))
logx2suncleancols1 =  pd.DataFrame('logx2sun'+ x2sun.columns.g
et_level_values(1)+ x2sun.columns.get_level_values(2))
logx2suncleancols = get_tokens(str(logx2suncleancols1))
logx2suncleancols = [x for x in logx2suncleancols if x.startsw
ith('logx2')]
logx2sun.columns =logx2suncleancols
x2sun.columns = x2suncleancols

x3sunprotection = x3.iloc[:, 45:48]
x3sunprotectioncleancols1 =  pd.DataFrame('x3sunprotection'+ x
3sunprotection.columns.get_level_values(1))
x3sunprotectioncleancols = get_tokens(str(x3sunprotectionclean
cols1))
```

```
x3sunprotectioncleancols = [x for x in x3sunprotectioncleancol
s if x.startswith('x3')]
logx3sunprotection = np.log(x3.iloc[:, 45:48].replace(0,1))
logx3sunprotectioncleancols1 =  pd.DataFrame('logx3sunprotecti
on'+ x3sunprotection.columns.get_level_values(1))
logx3sunprotectioncleancols = get_tokens(str(logx3sunprotectio
ncleancols1))
logx3sunprotectioncleancols = [x for x in logx3sunprotectioncl
eancols if x.startswith('logx3')]
logx3sunprotection.columns =logx3sunprotectioncleancols
x3sunprotection.columns = x3sunprotectioncleancols

x1sunaftersun = x1.iloc[:, 64:68]
x1sunaftersuncleancols1 =  pd.DataFrame('x1sunaftersun'+ x1sun
aftersun.columns.get_level_values(1))
x1sunaftersuncleancols = get_tokens(str(x1sunaftersuncleancols
1))
x1sunaftersuncleancols = [x for x in x1sunaftersuncleancols if
x.startswith('x1')]
logx1sunaftersun = np.log(x1.iloc[:, 64:68].replace(0,1))
logx1sunaftersuncleancols1 =  pd.DataFrame('logx1sunaftersun'+
x1sunaftersun.columns.get_level_values(1))
logx1sunaftersuncleancols = get_tokens(str(logx1sunaftersuncle
ancols1))
logx1sunaftersuncleancols = [x for x in logx1sunaftersuncleanc
ols if x.startswith('logx1')]
logx1sunaftersun.columns =logx1sunaftersuncleancols
x1sunaftersun.columns = x1sunaftersuncleancols
```

```
x3sunaftersun = x3.iloc[:, 48:51]
x3sunaftersuncleancols1 =  pd.DataFrame('x3sunaftersun'+ x3sun
aftersun.columns.get_level_values(1))
x3sunaftersuncleancols = get_tokens(str(x3sunaftersuncleancols
1))
x3sunaftersuncleancols = [x for x in x3sunaftersuncleancols if
x.startswith('x3')]
logx3sunaftersun = np.log(x3.iloc[:, 48:51].replace(0,1))
logx3sunaftersuncleancols1 =  pd.DataFrame('logx3sunaftersun'+
x3sunaftersun.columns.get_level_values(1))
logx3sunaftersuncleancols = get_tokens(str(logx3sunaftersuncle
ancols1))
logx3sunaftersuncleancols = [x for x in logx3sunaftersuncleanc
ols if x.startswith('logx3')]
logx3sunaftersun.columns =logx3sunaftersuncleancols
x3sunaftersun.columns = x3sunaftersuncleancols

x1lipcare = x1.iloc[:, 68:72]
x1lipcarecleancols1 =  pd.DataFrame('x1lipcare'+ x1lipcare.col
umns.get_level_values(1))
x1lipcarecleancols = get_tokens(str(x1lipcarecleancols1))
x1lipcarecleancols = [x for x in x1lipcarecleancols if x.start
swith('x1')]
logx1lipcare = np.log(x1.iloc[:, 68:72].replace(0,1))
logx1lipcarecleancols1 =  pd.DataFrame('logx1lipcare'+ x1lipca
re.columns.get_level_values(1))
logx1lipcarecleancols  =  get_tokens(str(logx1lipcarecleancols1
))
logx1lipcarecleancols = [x for x in logx1lipcarecleancols if x
```

```
.startswith('logx1')]
logx1lipcare.columns =logx1lipcarecleancols
x1lipcare.columns = x1lipcarecleancols

x2lipcare = x2.iloc[:, 126:136]
x2lipcarecleancols1 =  pd.DataFrame('x2lipcare'+ x2lipcare.col
umns.get_level_values(1)+  x2lipcare.columns.get_level_values(2
))
x2lipcarecleancols = get_tokens(str(x2lipcarecleancols1))
x2lipcarecleancols = [x for x in x2lipcarecleancols if x.start
swith('x2')]
logx2lipcare = np.log(x2.iloc[:, 126:136].replace(0,1))
logx2lipcarecleancols1 =  pd.DataFrame('logx2lipcare'+ x2lipca
re.columns.get_level_values(1)+  x2lipcare.columns.get_level_va
lues(2))
logx2lipcarecleancols  =  get_tokens(str(logx2lipcarecleancols1
))
logx2lipcarecleancols = [x for x in logx2lipcarecleancols if x
.startswith('logx2')]
logx2lipcare.columns =logx2lipcarecleancols
x2lipcare.columns = x2lipcarecleancols

x3lipcare = x3.iloc[:, 51:54]
x3lipcarecleancols1 =  pd.DataFrame('x3lipcare'+ x3lipcare.col
umns.get_level_values(1))
x3lipcarecleancols = get_tokens(str(x3lipcarecleancols1))
x3lipcarecleancols = [x for x in x3lipcarecleancols if x.start
swith('x3')]
logx3lipcare = np.log(x3.iloc[:, 51:54].replace(0,1))
```

```
logx3lipcarecleancols1 =  pd.DataFrame('logx3lipcare'+ x3lipca
re.columns.get_level_values(1))
logx3lipcarecleancols  =  get_tokens(str(logx3lipcarecleancols1
))
logx3lipcarecleancols = [x for x in logx3lipcarecleancols if x
.startswith('logx3')]
logx3lipcare.columns =logx3lipcarecleancols
x3lipcare.columns = x3lipcarecleancols

x1styling = x1.iloc[:, 72:76]
x1stylingcleancols1 =  pd.DataFrame('x1styling'+ x1styling.col
umns.get_level_values(1))
x1stylingcleancols = get_tokens(str(x1stylingcleancols1))
x1stylingcleancols = [x for x in x1stylingcleancols if x.start
swith('x1')]
logx1styling = np.log(x1.iloc[:, 72:76].replace(0,1))
logx1stylingcleancols1 =  pd.DataFrame('logx1styling'+ x1styli
ng.columns.get_level_values(1))
logx1stylingcleancols  =  get_tokens(str(logx1stylingcleancols1
))
logx1stylingcleancols = [x for x in logx1stylingcleancols if x
.startswith('logx1')]
logx1styling.columns =logx1stylingcleancols
x1styling.columns = x1stylingcleancols

x3styling = x3.iloc[:, 54:57]
x3stylingcleancols1 =  pd.DataFrame('x3styling'+ x3styling.col
umns.get_level_values(1))
x3stylingcleancols = get_tokens(str(x3stylingcleancols1))
```

```
x3stylingcleancols = [x for x in x3stylingcleancols if x.start
swith('x3')]
logx3styling = np.log(x3.iloc[:, 54:57].replace(0,1))
logx3stylingcleancols1 =  pd.DataFrame('logx3styling'+ x3styli
ng.columns.get_level_values(1))
logx3stylingcleancols  =  get_tokens(str(logx3stylingcleancols1
))
logx3stylingcleancols = [x for x in logx3stylingcleancols if x
.startswith('logx3')]
logx3styling.columns =logx3stylingcleancols
x3styling.columns = x3stylingcleancols

x1haircare = x1.iloc[:, 76:80]
x1haircarecleancols1 =  pd.DataFrame('x1haircare'+ x1haircare.
columns.get_level_values(1))
x1haircarecleancols = get_tokens(str(x1haircarecleancols1))
x1haircarecleancols = [x for x in x1haircarecleancols if x.sta
rtswith('x1')]
logx1haircare = np.log(x1.iloc[:, 76:80].replace(0,1))
logx1haircarecleancols1 =  pd.DataFrame('logx1haircare'+ x1hai
rcare.columns.get_level_values(1))
logx1haircarecleancols  =  get_tokens(str(logx1haircarecleancols
1))
logx1haircarecleancols = [x for x in logx1haircarecleancols if
x.startswith('logx1')]
logx1haircare.columns =logx1haircarecleancols
x1haircare.columns = x1haircarecleancols

x3haircare = x3.iloc[:, 57:61]
```

```
x3haircarecleancols1 =  pd.DataFrame('x3haircare'+ x3haircare.
columns.get_level_values(1))
x3haircarecleancols = get_tokens(str(x3haircarecleancols1))
x3haircarecleancols = [x for x in x3haircarecleancols if x.sta
rtswith('x3')]
logx3haircare = np.log(x3.iloc[:, 57:61].replace(0,1))
logx3haircarecleancols1 =  pd.DataFrame('logx3haircare'+ x3hai
rcare.columns.get_level_values(1))
logx3haircarecleancols = get_tokens(str(logx3haircarecleancols
1))
logx3haircarecleancols = [x for x in logx3haircarecleancols if
x.startswith('logx3')]
logx3haircare.columns =logx3haircarecleancols
x3haircare.columns = x3haircarecleancols

x5aftersun = x5.iloc[:, 0:4]
x5labello = x5.iloc[:, 36:56]
x5bath = x5.iloc[:, 57:68]
x5shower = x5.iloc[:, 69:100]
x5inshower = x5.iloc[:, 146:153]
x5soap = x5.iloc[:,101:116]
x5bodyessentials = x5.iloc[:, 117:145]
x5bodyperformance = x5.iloc[:,154:166]
x5bodyapc = x5.iloc[:,167:181]
x5facecare = x5.iloc[:, 182:244]
x5facecleansing = x5.iloc[:,245:289]
x5hand = x5.iloc[:,290:302]
x5deofemale = x5.iloc[:, 302:339]
x5deomale = x5.iloc[:,340:367]
```

```
x5haircare = x5.iloc[:, 368:401]
x5styling = x5.iloc[:, 402:419]
x5aftershave = x5.iloc[:, 420:431]
x5menfacecare = x5.iloc[:, 432:465]
x5sunprotect = x5.iloc[:,466:514]

# Data Cleaning
def datacleaning(category, list):
    '''
    A function that will remove the products for which there i
s not enough data to perform any analysis
    Thee are either already discontinued or have been added to
the market less than 4 years ago.
    The function takes two parameters: the category from which
the products will be removed and the list
    of products that need to be removed, which come from a gra
phical analysis
    '''
    name = category
    category_clean = category.drop(list, axis = 1, level = 4)
    return category_clean

aftersun_pd_clean = datacleaning(aftersun_pd, ['NSUN_IN-SHWR_R
FRSHNG_AFT_SUN_LTN_250ML','NSUN_AFTR_SUN_BRNZ&TAN_PROL._LTN_20
0ML'])
labello_pd_clean = datacleaning(labello_pd, ['LAB_CR&CLR_ND_4,
8G', 'LAB_PCH_4,8G', 'LAB_CHERRY,_4,8_GR', 'LAB_STRAWBERRY,_4,
8_GR',    'LAB_FR_SHN_PNK_WTR_MLN_4,8G','LAB_FRU_SHN_BLB_4,8G',
'LAB_VNL_BTR_CRM_4,8G', 'LAB_CR_BL_RPBY_RD_APL_7G', 'LAB_CR_BL
```

```
_FRS_MNT_7G','LAB_FOR_MEN,_4,8_GR', 'LAB_LP_BTR_TIN_ORG_16,7G'
,'LAB_LP_BTR_TIN_RPBY_RS_16,7G', 'LAB_LP_BTR_TIN_VNL_16,7G',
'LAB_MED_PROTECTION,_4,8_GR', 'LAB_SOS_LP_RPR_6ML', 'LAB_MIXED
_DISPLAY', 'LAB_MILK&HONEY,_4,8_GR', 'LAB_SUN_SPF25,_4,8_GR'])
bath_pd_clean = datacleaning(bath_pd, ['NBC_SHR_CRM_ALO_750ML'
,'NBC_BTH_SPR_TCH_500ML', 'NBC_BTH_CREME_SOFT_750ML','NBC_BTH_
CRM_SMT_750ML', 'NBC_BTH_RELAXING_MOMENTS_500ML','NBC_BTH_CARE
_&_RELAX_750ML','NBC_BTH_GD_BYE_STR_750ML', 'NBC_BTH_DRM_SNS_7
50ML', 'NBC_BTH_WLC_SUN_SHN_750ML'])
shower_pd_clean = datacleaning(shower_pd, ['NBC_SHR_CLY_FRS_GN
G&BSL_500ML', 'NBC_SHR_CLY_FRS_HBS&WSAG_500ML', 'NBC_SHR_CARE_
&_COCOA__500ML','NBC_SHR_CARE_&_COCONUT_500ML', 'NBC_SHR_CR&CC
N_750ML','NBC_SHR_CRM_CR_250ML', 'NBC_SHR_SPR_TCH_500ML','NBC_
SHR_CRM_SFT_750ML', 'NBC_CRM&OIL_PRS_CHR_BLS_500ML','NBC_CRM&O
IL_PRS_LTS_500ML','NBC_SHR_CARE_&_ROSES_500ML', 'NBC_SHR_CARE_
&_STARFRUIT_500ML','NBC_SHR_MEN_DP_500ML', 'NBC_SHR_MINI_ENERG
Y_FOR_MEN_50ML','NBC_SHR_PWR_FRT_FRS_500ML', 'NBC_SHR_PUR_FRS_
500ML', 'NBC_SHR_SLK_MOE_CRM_CR_200ML','NBC_SHR_SLK_MOE_SMR_01
_200ML', 'NBC_SHR_SLK_MOE_VNL_CRM_200ML','NBC_SHR_MUSCLE_RELAX
_250ML', 'NBC_SHR_PROTECT_&_CARE_500ML','NBC_SHR_PUR_FOR_MEN_2
50ML', 'NBC_SHR_MEN_RCK_SAL_250ML','NBC_SHR_CR&SUN_SHN_250ML',
'NBC_SHR_SUN_SHN_LV_500ML','NBC_SHR_WATER_LILY_&_OIL_250ML'])
inshower_pd_clean = datacleaning(inshower_pd,['NBODY_ESS_IN_SH
R_LTN_250ML','NBODY_ESS_IN_SHR_MLK_80ML', 'NBODY_ESS_IN_SHR_BT
Y_GLW_250ML', 'NBODY_ESS_IN_SHR_HNY_MLK_250ML'])
soap_pd_clean = datacleaning(soap_pd,['NBC_SOAP_FP_COCONUT_90G
R', 'NBC_SOAP_LS_DMN_BTY_250ML','NBC_LQD_SOP_REF_DMN_TCH_500M
L', 'NBC_SOAP_FB_HONEY_&_OIL_100GR','NBC_LQD_SOP_HONEY&OIL_250
ML', 'NBC_LQD_SOP_HONEY&OIL_500ML','NBC_LQD_SOP_CRM_CR_250ML',
```

```
'NBC_BAR_SOP_CRM_CR_BOX_3X100GR', 'NBC_SOAP_FB_HAPPYTIME_3X100
GR','NBC_LQD_SOP_LMN&OIL_250ML', 'NBC_SOAP_FP_MILK_90GR','NBC_
SOAP_FP_MIXED_DISPLAY'])
bodyessentials_pd_clean = datacleaning(bodyessentials_pd, ['NB
ODY_RCH_CRG_MLK_250ML', 'NBODY_EXP_HYD_LTN_250ML', 'NBODY_EXP_
HYD_LTN_400ML', 'NBODY_MIXED_DISPLAY', 'NBODY_RCH_CRG_MLK_400M
L','NBODY_ESS_MSE_NRS_MLK_200ML', 'NBODY_MIXED_DISPLAY','NBODY
_MIXED_DISPLAY', 'NBODY_ESS_ENR_LTN_200ML','NBODY_ESS_RLX_LTN_
200ML', 'NBODY_ESS_SNSL_LTN_200ML','NBODY_ESS_ORG_BLSM_BQT_200
ML', 'NBODY_ESS_MSE_WLD_RSBY&WHT_TEA_200ML','NBODY_ESS_CCN_MN_
OIL_CRM_200ML', 'NBODY_ESS_CHRY_BLSM__JJB_CRM_200ML','NBODY_ES
S_MSE_CCMB_SPL&MTCH_TEA_200ML', 'NBODY_ESS_VNL_ALMN_OIL_200ML'
,'NBODY_ESS_CHRY_BLSM__JJB_OIL_200ML', 'NBODY_ESS_CC__MCDM_OIL
_200ML','NBODY_ESS_RSE__ARG_OIL_200ML', 'NBODY_PUR&NTR_MLK_250
ML','NBODY_RPR&CR_LTN_300ML__PMP', 'NBODY_SNS_LTN_250ML', 'NBO
DY_SMT_MLK_400ML', 'NBODY_ESS_MSE_SMT_SNS_200ML','NBODY_IN_SHR
_LTN_Q10_250ML'])
bodyperformance_pd_clean = datacleaning(bodyperformance_pd, [
'NBODY_Q10_MLK_400ML','NBODY_FIRMING_OIL_200ML','NBODY_FIR_LTN
_Q10+_400ML', 'NBODY_Q10_CRM_300_ML','NBODY_PRFR_Q10_LGG_S-M_1
PCS', 'NBODY_PRFR_Q10_LGG_L-XL_1PCS', 'NBODY_PRFR_Q10_+_FRSH_L
GS_200ML','NBODY_PRFR_Q10_PNT_S-M_1PCS', 'NBODY_PRFR_Q10_PNT_L
-XL_1PCS'])
bodyapc_pd_clean = datacleaning(bodyapc_pd, ['NCR_NRS_CRM_200M
L', 'NCR_NRS_CRM_400ML', 'NCR_SNS_CRM_200ML', 'NSFT_PROMOTION'
,'N_SFT_PLS_JAR_LMT_EDT_GRN_100ML', 'N_SFT_PLS_JAR_LMT_EDT_PNK
_100ML', 'N_SFT_PLS_JAR_LMT_EDT_YLW_100ML', 'NSFT_TUBE_75_ML',
'NSFT_JAR_300_ML'])
facecare_pd_clean = datacleaning(facecare_pd, ['NF_CLL_AA_DY_C
```

```
R_SPF30_50ML','NF_CLL_MAT_DY_CRM_SPF30_50ML', 'NF_CLL_MAT_NGT_
CRM_50ML','NF_CLL_MAT_OIL_SRM_30ML', 'NFC_CLL_MAT_AGE_SPT_30M
L','NFC_CLL_VOL_SHEET_MSK', 'NF_CLL_RDNCE_FLUID_SPF15_40ML','N
F_CLL_RDNCE_EYE_LIGHT_15ML', 'NF_CLL_RDNCE_NGHT_ESSNCE_40ML',
'NF_CLL_AA_SRM_PEARLS_30ML', 'NF_CLL_AA_VOLUME_FLLNG_DAY_50ML'
,'NF_CLL_VLM_FLLNG__SRM_AMP_5ML', 'NF_ESS_NIGHT_NRML_JAR_50ML'
, 'NF_DAY_N/M_UV_PROTECT._TUBE_50ML', 'NF_ESLS_DY_CRM_N/M_SPF3
0_50ML_JAR','NF_ESLS_DY_D/S_SPF30_50ML_JAR', 'NF_PURE_CONTROL_
SHINE_TUBE_75ML','NF_ESLS_SNS_DY_SPF15_50ML_JAR', 'NF_ESLS_SNS
_NGT_50ML_JAR','NF_ESS_MOISTURISING_2X7,5_ML', 'NF_ESS_BLMSH_P
RONE_MSTRSR_50ML_TUBE','NF_ESS_BLMSH_PRONE_BB_CRM_50ML_TUBE',
'NF_ESS_BLMSH_PRONE_CNTRL_SHN_DAY_50ML_T', 'NF_DNAGE_NIGHT_50M
L_JAR','NF_P&N_ANTI_AGE_DAY_CRM,_50_ML_JAR','NF_P&N_ANTI_AGE_N
GHT_CRM,_50_ML_JAR', 'NF_MIXED_DISPLAY', 'NF_Q10_MASK_2X7,5ML'
, 'NFC_Q10_CUSH_15ML_MEDIUM','NFC_Q10_CUSH_15ML_DARK', 'NFC_Q1
0_NGHT_CRM_VIT_C_40ML','NFC_Q10_DAYCARE_20ML_FB', 'NF_Q10_EYE_
ROLL-ON_10_ML','NF_Q10_DAYCARE_NOUR,_50ML', 'NF_Q10_OIL,_30ML'
,'NF_Q10_CC_CREAM_50ML_TUBE', 'NFC_Q10_C_SHEET_MSK','NF_Q10_DY
_SPF30_XTR_PRTCTN_50ML', 'NF_Q10_SERUM_PEARLS_40ML_DISP', 'NF_
ESLS_MASK_MOISTURIZING_75ML_TUBE','NFC_ESS_URBAN_SKIN_DAY_CARE
_50ML','NFC_ESS_URBAN_SKIN_NIGHT_CARE_50ML','NF_VIT_XTR_NOURSH
NG_DAY_JAR_50ML', 'NF_VIT_CALC_DAY_50ML_JAR','NF_VIT_SOY_EYE_1
5ML_TUBE', 'NF_VIT_SOY_DAY,_JAR_50_ML', 'NF_VIT_SOY_NIGHT_50ML
_JAR', 'NF_CLL_AA_SRM_40ML'])
facecleansing_pd_clean = datacleaning(facecleansing_pd, ['NF_E
SLS_BI-PHASE_EMR_BLUE_125ML', 'NF_MIXED_DISPLAY','NF_ESLS_PEEL
ING_NRML_75ML', 'NF_P&N_MILK_ALL_SKN_TPS_BTL_200ML','NF_ESLS_E
XFOLIATING_MASK_2X7,5ML','NF_ESS_BLMSH_PRONE_3IN1_150ML', 'NF_
ESLS_PMP_WSH_OIL_NRML_150ML','NF_ESLS_PMP_WSH_OIL_DRY_150ML',
```

```
'NF_ESLS_RC_SCRB_CMBNTN_75ML','NF_ESS_INSHWR_MUR_NRML_150ML',
'NF_ESS_INSHWR_MUR_SNSTV_150ML_TOTTLE','NF_ESLS_CLN_MOUSSE_DRY
_DISP_150ML', 'NF_ESS_CRM_CARE_MILK_BTL_200ML','NF_ESS_CRM_CAR
E_WSH_CRM_TUBE_150ML', 'NF_MCLLR__WTR_IN_OIL_RS_400ML','NF_ESL
S_MCLLR_WTR_SENS_200ML', 'NF_ESLS_MCLLR_WTR_SENS_100ML','NF_ES
LS_MCLLR_EXP_MUR_WTPRF_400ML', 'NF_ESLS_MCLLR_WTR_NRML_400ML',
'NF_ESLS_MCLLR_WTR_SENS_400ML','NF_VIT_MILK_BTL_200_ML', 'NF_V
IT_TONER,_200_ML_BOTTLE','NF_DTX_CLAY_WASH_150ML', 'NF_DTX_SHE
ET_MASK_1PC','NF_DTX_PORE_REFINE_MASK_75ML', 'NF_DTX_PEEL_OFF_
MASK_75ML','NF_DTX_PURIFY_MASK_75ML', 'NF_DTX_PEEL_OFF_MASK_10
ML','NF_DTX_MCLLR_WTR_400ML', 'NF_ESLS_WIPES_NRML_25PCS','NF_E
SLS_WIPES_DRY_25PCS', 'NF_ESLS_WIPES_NRML_25X2_MULTIPACK','NF_
ESLS_WIPES_DRY_25X2_MUL', 'NF_ESLS_MCLLR_WIPES_25X2_MULTIPA',
'NF_MCLLR_WPS_RS_25PCS', 'NF_ESLS_WPS_BIODGRDBL_25PCS','NF_ESL
S_WPS_NRML_40PCS', 'NF_ESLS_WIPES_DRY_40PCS','NF_ESS_CRM_CARE_
WIPES_25PCS'])
deofemale_pd_clean = datacleaning(deofemale_pd, ['NDEO_SPY_INV
_BLC&WHT_PUR_FM_150ML','NDEO_SPY_MINI_INV_BLC&WHT_CLR_FM_35ML'
,'NDEO_RLL-ON_INV_BLC&WHT_PUR_FM_50ML','NDEO_SPY_INV_BLC&WHT_S
LK_SMTH_FML_150ML','NDEO_RLL-ON_INV_BLC&WHT_SLK_SMTH_FML_50M',
'NDEO_RLL-ON_BLC&WHT_FRSH__FML_50ML','NDEO_DEO_SPY_INV_BLC&WHT
_FRSH__FML_150ML','NDEO_RLL-ON_CLM&CR_FML_50ML', 'NDEO_RLL-ON_
DBL_EFF_VLT_SNS_F_50ML','NDEO_SPY_DMLK_SNS_AP_FML_150ML','NDEO
_RLL-ON_DEO_MLK_PINK_AP_FML_40ML','NDEO_SPY_DEO_MLK_BLUE_AP_FM
L_150ML','NDEO_RLL-ON_DEO_MLK_BLUE_AP_FML_40ML', 'NDEO_RLL-ON_
DRY_LE__FM_50ML','NDEO_SPY_DRY_LE_FM_200ML', 'NDEO_SPY_FRS_FLW
_FML_150ML','NDEO_RLL-ON_FRS_FLW_FML_50ML_GLASS_50ML', 'NDEO_R
LL-ON_FRESH_F_50ML','NDEO_SPY_FRESH_AP_F_200ML', 'NDEO_SPY_PRO
TECT_&_CARE_AP_FML_200ML','NDEO_RLL-ON_PROTECT_&_CARE_AP_FML_5
```

```
0ML',    'NDEO_SPY_PRL&BTY_F_200ML','NDEO_RLL-ON_PRL&BTY_50ML',
'NDEO_SPY_PURE_F_200ML','NDEO_RLL-ON_PURE_F_50ML',  'NDEO_RLL-O
N_POWDER_TOUCH_FML_GLAS_50ML','NDEO_SPY_POWDER_TOUCH_FML_150M
L',   'NDEO_SPY_PUR&SNS_FML_200ML','NDEO_RLL-ON_PUR&SNS_FML_50M
L',   'NDEO_SPY_STR_PRT_FML_150ML','NDEO_RLL-ON_STR_PRT_FML_50M
L'])
deomale_pd_clean = datacleaning(deomale_pd,['NDEO_RLL-ON_BLC&W
HT_FRSH_ML_50ML',  'DEO_SPY_BLC&WHT_FRSH__ML_150ML','NDEO_SPY_B
LACK_MALE_150ML',  'NDEO_RLL-ON_BLACK_MALE_50ML','NDEO_RLL_ON_D
P_BRW_50ML_ML',  'NDEO_SPY_DP_BRW_150ML_ML','NDEO_RLL-ON_DRY_LE
__ML_50ML',  'NDEO_SPY_DRY_LE_ML_200ML','NDEO_SPY_FRESH_ACT_ML_
200ML',  'NDEO_RLL-ON_FRESH_M_50ML','NDEO_SPY_FRS_OCN_ML_150ML'
,  'NDEO_RLL-ON_FRS_OCN_ML_GLASS_50ML','NDEO_SPY_PROTECT_&_CARE
_AP_ML_200ML','NDEO_RLL-ON_PROTECT_&_CARE_AP_ML_50ML',  'NDEO_S
PY_SNS_PRTKT_ML_150ML','NDEO_SPY_SNS_PRTKT_ML_200ML','NDEO_RLL
-ON_SLV_PRTT_M_50ML',  'NDEO_SPY_SPORT_M_150ML','NDEO_RLL-ON_ST
R_PRT_ML_50ML'])
haircare_pd_clean = datacleaning(haircare_pd, ['NHC_SHM_ADD_GL
S_250ML','NHC_SHM_NM_ADD_PR_250ML',  'NHC_SHM_NM_ADD_PWR_250ML'
,'NHC_SHM_BLNC_FRS_CR_400ML',  'NHC_SHM_NM_CL_FRSH_250ML','NHC_
SHM_COL_CR_PRT_400ML',  'NHC_CNDR_COL_CR_PRT_200ML','NHC_SHM_CL
SC_MLD_CR_400ML',  'NM_SHM_BLCK_ML_250ML','NHC_SHM_DMN_GLS_CR_4
00ML',  'NHC_CNDR_DMN_GLS_CR_200ML','NHC_SHM_HR_MLK_250ML',  'NH
C_CNDR_HR_MILK_200ML','NHC_SHM_HR_MLK_400ML',  'NHC_LV_IN_CNDRP
MP_HRMLK_NORMAL_DRY_200ML','NHC_CNDR_HAIRMILK_THICK_200ML',  'N
HC_SHM_HAIRMILK_THICK_400ML','NHC_SHM_INT_CR_RPR_400ML',  'NHC_
CNDR_INT_CR_RPR_200ML','NHC_SHM_LNG_RPR_400ML',  'NHC_CNDR_LNG_
CR_RPR_200ML','NHC_SHM__MCL_MSG_400ML',  'NHC_SHM_MCL_GRS_HR_SC
P_400ML','NHC_SHM_MCL_SNS_HR_SCP_400ML',  'NHC_SHM_MCL_COL_SCR_
```

```
400ML','NHC_CNDR_MCL_CLR_SCR_200ML', 'NHC_SHM_PRT_CR_250ML','N
HC_CNDR_RPR_TRGT_CR_200ML', 'NHC_SHM_RPR_TRGT_CR_400ML','NHC_S
HM_SP_250ML', 'NHC_SHM_NM_STRG_PWR_250ML','NHC_SHM_VLM_CR_400M
L'])
styling_pd_clean = datacleaning(styling_pd, ['NHS_SPY_COL_CR_P
RT_250ML','NHC_STYLING_SPRAY_CARE&HOLD', 'NHC_STYLING_MOUSSE_C
ARE&HOLD','NHS_MOE_EXT_SRG_HLD_150ML','NHS_BLM_FLX_CRL_CR_150M
L','NHC_STYL_GEL_MTTFYNG_SHP_150ML','NHC_STYL_GEL_DFN_SHN_150M
L', 'NHS_SPY_SRG_HLD_250ML','NHS_SPY_VITAL_LQU_250ML', 'NHS_SP
Y_VLM_CR_250ML'])
aftershave_pd_clean = datacleaning(aftershave_pd, ['NM_AS_LTN_
BLACK_100ML', 'NM_BDY_PRTCT&CR_AS_LTN_240ML','NM_AS_LTN_PRTCT&
CARE_2-PHASE_100ML', 'NM_AS_BLM_SNSTV_COOLNG_100ML','NM_SNSTV_
BLM_SKNSTBL_125ML', 'NM_AS_LTN_ACTV_NRGY_100ML','NM_AS_BLM_ACT
V_NRGY_100ML'])
menfacecare_pd_clean = datacleaning(menfacecare_pd,['NMEN_BRD_
SKN_WSH_100ML', 'NM_FCL_GEL_WASH_PRTCT&CR_100ML','NM_FC_CRM_AC
TIVE_AGE_DNAGE_50ML', 'NM_FC_CRM_AP_150ML', 'NM_FC_CRM_AP_30M
L', 'NIVEA_MEN_BEARD_OIL_75ML','NMEN_BRD_SKN_GEL_50ML','NM_FC_
GEL_SNSTV_HYDRO_50ML', 'NM_FC_CRM_SNSTV_COOLNG_MOIST_50ML','NM
_FC_CRM_SNSTV_SPF15_50ML','NM_FC_GEL_ACTV_NRGY_MRNG_FX_50ML',
'NM_FC_EYE_ROLL_ON_SKN_NRGY_10ML','NM_SH_GEL_BLACK_200ML','NM_
SH_FOAM_PRTCT&CR_250ML', 'NM_BDY_PRTCT&CR_SH_STCK_75ML','NMEN_
BDY_PRCTC&CR_SHG_GEL_200ML', 'NM_SH_FOAM_SNSTV_200ML', 'NM_SH_
GEL_SNSTV_ONESTROKE_200ML', 'NM_SH_FOAM_SNSTV_MINI_35ML','NM_S
H_GEL_SNSTV_MINI_30ML', 'NM_SH_GEL_SNSTV_COOLNG_200ML','NM_SH_
FOAM_SLV_PRTCT_200ML', 'NM_SH_GEL_ACTV_NRGY_200ML'])
sunprotect_pd_clean = datacleaning(sunprotect_pd, ['NSUN_DEEP_
TAN_OIL_SPF_6,_200ML','NSUN_DT_OIL_SPY_SPF6,_150ML', 'NSUN_DT_
```

```
OIL_200ML','NSUN_FC_CRM_PRTT&BRN_SPF50_50ML','NSUN_UV_FACE_SEN
SITIVE_CRÈME_SPF_50,_50M','NSUN_FC_CRM_SHN_CTRL_SPF30_50ML','N
SUN_FC_CRM_A_A&PGMT_BB_SPF50_50ML','NSUN_FC_ANTI-AGE&PIGM_CRM_
SPF_50_50ML','NSUN_KIDS_ROLL-ON_PINK_50+,_50ML',  'NSUN_KIDS_RO
LL-ON_GREEN_50+,_50ML','NSUN_KIDS_SENSITIVE_TRIGGER_SPRAY_300M
L','NSUN_KIDS_SWM&PLY_PROT_LTN_SPF50+,150ML','NSUN_KIDS_PRT&SE
NS_ROLL-ON_SPF_50+_50ML','NSUN_KIDS_PRTCT_LTN_ROLL-ON_SPF50+_5
0ML','NSUN_BABY_PROT_LTN_SPF_50+,_200ML','NSUN_P&B_ACTVT_PRT_O
IL_SPRY_SPF_20_200ML','NSUN_P&B_ACTVT_PRT_OIL_SPRY_SPF_30_200M
L','NSUN_PRT_&_BRN_TRG_SPY_SPF_30,_300ML','NSUN_SPY_SPF_20,_20
0ML',  'NSUN_LTN_SPF_30,_200ML','NSUN_LTN_SPF_30,_400ML','NSUN_
TRIGGER_SPRAY_50+,_300ML','NSUN_PRTCT&MSTR_LTN_SPF_50+_400ML',
'NSUN_PRTCT_&_MSTR_ROLL-ON_SPF_50+_50ML','NSUN_PRTCT&RFRSH_CLN
G_MIST_SPF_30_200ML','NSUN_PRTCT&RFRSH_CLNG_MIST_SPF_20_200ML'
,'NSUN_INVS_PROT_SPY_SPF20,_200ML',  'NSUN_SENS_SOOTH_LOTION_SP
F_50+200M_L','NSUN_SENSITIVE_SOOTHING_SPRAY_50+,_200ML','NSU_P
URE_&_SENSITIVE_SPY_SPF30_200ML','NSU_PURE_&_SENSITIVE_SPY_SPF
50_200ML'])

y = pd.concat([aftersun_pd_clean, labello_pd_clean, bath_pd_cl
ean, shower_pd_clean, inshower_pd_clean, soap_pd_clean, bodyes
sentials_pd_clean, bodyperformance_pd_clean, bodyapc_pd_clean,
facecare_pd_clean, facecleansing_pd_clean, deofemale_pd_clean,
deomale_pd_clean, haircare_pd_clean, styling_pd_clean, aftersh
ave_pd_clean, menfacecare_pd_clean, sunprotect_pd_clean], join
= 'outer', axis = 1, ignore_index = False)
category_names = pd.DataFrame(['AFTER_SUN', 'LABELLO', 'NBC_BA
TH', 'NBC_SHOWER', 'NBC_SOAP', 'NBODY_ESSENTIAL', 'NBODY_INSHO
WER', 'NBODY_PERFORMANCE', 'NCRM_CREME', 'NCRM_SOFT', 'NF_CAR
```

```
E', 'NF_CLEANSE', 'NIVEA_DEO_FEMALE', 'NIVEA_DEO_MALE', 'NIVEA
_HAIR_CARE', 'NIVEA_HAIR_STYLING', 'NM_AFTER_SHAVE', 'NM_MOIST
URISER', 'NM_SHAVING', 'PROTECTION'], columns =['Categories'])

# Plot the Times series of each category

def timeseriesplot(data, m, n, category):
    """Plot the time series of each product. Takes as parameters a data frame c composed by all the products in category c,
    m, n - the number of rows and columns the figure should have.
    """

    fig_data = plt.figure(figsize = (n, m))
     fig_data.subplots_adjust(hspace = 0.3, wspace = 0.2, top=0.9)
     plt.suptitle(data.iloc[:,[0]].columns.get_level_values(0)[0] + ' sales of category '+ category)
    size = len(data.columns)
    for i in range(int(size)):
        m1 = m-2
        n1 = n-2
        fig_i = fig_data.add_subplot(m1, n1, i+1)
         graph_i = fig_i.plot(np.array(data.iloc[:,[i]]), color = 'navy', linewidth = 0.7)
         fig_i.set_title(data.iloc[:,[i]].columns.get_level_values(3)[0], fontsize = 'xx-small')
        fig_i.set_xticklabels([])
        fig_i.set_yticklabels([])
```

```
        fig_data.savefig(data.iloc[:,[0]].columns.get_level_values(0)[0] + '_sales_of_category_'+ data.iloc[:,[0]].columns.get_level_values(1)[0]+ '.png')

def plottimeseries():
    category = input('Enter the category you want to analyse:')
    data = y.xs(category, level = 1, axis = 1)
    n = int(len(data.columns)+3)
    return timeseriesplot(data, 4, n, category)
# Multiple Regression Model

def transformdata(yc, x11, logx11, x21, logx21, x31, logx31, x51):
    for p in range(len(yc.columns)):
        yp = pd.DataFrame(yc.iloc[:, p])
        yp.columns=['y']
        x51 = pd.DataFrame(x51.iloc[:, [p]])
        x51.columns = ['x5']
        logyp = pd.DataFrame(np.log(yc.iloc[:, p].replace(0, 1)))
        logyp.columns=['logy']
        logx4 = np.log(x4.replace(0,1))
        logx51 = pd.DataFrame(np.log(x51.iloc[:, p].replace(0, 1)))
        logx51.columns = ['logx5']
        loge_n = np.log(e_n.replace(0,1))
        linlindata = pd.concat([yp, x11, x21, x31, x4, x51, e_n], axis = 1, join = 'outer', ignore_index = False)
```

```
        print(str(yc.iloc[:,[p]].columns.get_level_values(4)[0
]))
        loglindata = pd.concat([logyp, x11, x21, x31, x4, x51,
e_n], axis = 1, join = 'outer', ignore_index = False)
        print(str(yc.iloc[:,[p]].columns.get_level_values(4)[0
]))
        linlogdata = pd.concat([yp, logx11, logx21, logx31, lo
gx4, logx51, loge_n], axis = 1, join = 'outer', ignore_index =
False)
        print(str(yc.iloc[:,[p]].columns.get_level_values(4)[0
]))
        loglogdata = pd.concat([logyp, logx11, logx21, logx31,
logx4, logx51, loge_n], axis = 1, join = 'outer', ignore_index
= False)
        print(str(yc.iloc[:,[p]].columns.get_level_values(4)[0
]))
         data = pd.concat([linlindata, loglindata, linlogdata,
loglogdata], keys = ['linlin', 'loglin', 'linlog', 'loglog'],
axis = 1, join = 'outer', ignore_index = False)
        return data
        p+=1
#Use forward stepwise strategy to find the linear model that m
aximizes the adjusted r sqaured

def forward_selected(data, response):
    """Linear model designed by forward selection.
    Parameters:
    -----------
    data : pandas DataFrame with all possible predictors and r
```

```
esponse
    response: string, name of response column in data
    Returns:
    --------
    model: an "optimal" fitted statsmodels linear model
           with an intercept
           selected by forward selection
           evaluated by adjusted R-squared
    """
    remaining = set(data.columns)
    remaining.remove(response)
    selected = []
    current_score, best_new_score = 0.0, 0.0
    while remaining and current_score == best_new_score:
        scores_with_candidates = []
        for candidate in remaining:
            formula = "{} ~ {} + 1".format(response,
                                           ' + '.join(str(s) f
or s in selected + [candidate]))
            score = smf.ols(formula, data).fit().rsquared_adj
            scores_with_candidates.append((score, candidate))
        scores_with_candidates.sort()
        best_new_score, best_candidate = scores_with_candidate
s.pop()
        if current_score < best_new_score:
            remaining.remove(best_candidate)
            selected.append(best_candidate)
            current_score = best_new_score
    formula = "{} ~ {} + 1".format(response,
```

```
                                        ' + '.join(str(s) for s in
selected))
    model = smf.ols(formula, data).fit()
    return model

# This function returns the best model among the 4 possible tr
ansformations
def bestrmodel(models):
    """Returns the best model (based on the adjusted r square
d) to describe the slaes of product p,
    takes as a parameter a list of the best model of each tran
sformation
    """
    adjr = [model1.rsquared_adj, model2.rsquared_adj, model3.r
squared_adj, model4.rsquared_adj]
    bestmodel = models[adjr.index(max(adjr))]
    return bestmodel

def residuals(model):
    """Resturns the residuals plot against the fitted values,
 as well as the histogram of residuals.
    Takes as parameters a model and the number of rows and col
umns the figure must have
    """
    fig_res = plt.figure(figsize= (8, 3))
    fig_res.subplots_adjust(hspace = 0.3, wspace = 0.2, top=0.
9)
    res = fig_res.add_subplot(1,2,1)
    res.plot(model.fittedvalues, model.resid, linestyle='', ma
```

```
rker = 'o', color = 'navy')
    res.set_xticklabels([])
    res.set_yticklabels([])
     res.set_title('Residuals against fitted values', fontsize
= 'x-small')
    hist = fig_res.add_subplot(1,2,2)
    hist.hist(model.resid, histtype = 'bar', color = 'navy')
    hist.set_yticklabels([])
    hist.set_xticklabels([])
     hist.set_title('Histogram of Residuals', fontsize = 'x-sma
ll')
    fig_res.savefig('Hist_res.png')

# Function to see if the model is statistically significant

def modelsignificance(model):
    if (model.f_pvalue < 0.05):
         print(model.f_pvalue, "We can reject the null hypothes
is and conclude that the model is overall significant")
    else:
         print("We are not able to reject the null hypothesis,
 hence we can't conclude that the model is significant")

# Model each variable with a high p-value against all the othe
rs
def multicolcheck(model, x, data):
    l = []
    for i in range(len(model.params.index)-1):
        l.append(i+1)
```

```
    params = data[model.params.index[l]]
    Xparams = params.drop(labels = x, axis =1)
    model = sm.OLS(data[x], Xparams).fit()
    return model

def VIF(model):
    VIF = 1/(1-model.rsquared)
    return VIF

def computeVIF(model, data):
    listvariables = model.params.index.drop('Intercept')
    vifs = []
    names = []
    for i in range(len(listvariables)):
        VIF1 = VIF(multicolcheck(model, listvariables[i], data
)).round(2)
        vifs1 = {VIF1: listvariables[i]}
        vifs2 = {listvariables[i]: VIF1}
        vifs.append(list(vifs1))
        names.append(list(vifs2))
    datavifs = [vifs, names]
    vifstable = np.transpose(pd.DataFrame(datavifs, index = [
'VIF', 'Variable']))
    print('Máximum VIF is:', max(vifstable['VIF']))
    return vifstable


# Remove the redundant variable from the model
def newmodel(model, x, data, response):
```

```
    l = []
    for i in range(len(model.params.index)-1):
        l.append(i+1)
    params = data[model.params.index[l]]
    Xparams = params.drop(labels = x, axis =1)
    formula = "{} ~ {} + 1".format(response,
                                   ' + '.join(str(s) for s in
Xparams))
    newmodel = smf.ols(formula, data=data).fit()
    return newmodel

# Def to check if the time series of product p - after observi
ng the time series graph, this test will be applied
# to time series suggesting stationarity. A confidence level o
f 95% will be used.
# Hence if p-value > 0.05 H0 is not rejected and data is not s
tationary; otherwise H0 is rejected and the data is stationary
# H0: The time series is not stationary
# H1: The time series is stationary
def stationarity(yc):
    from statsmodels.tsa.stattools import adfuller
    size = len(yc.columns)
    for p in range(size):
        yp = yc.iloc[:, p]
        result = adfuller(yp.dropna())
        if (result[1] < 0.95):
            x= print('ADF Statistic: %f' % result[0],'\n','p-v
alue: %f' % result[1], '\n', yc.iloc[:,[p]].columns.get_level_
values(4)[0] + ' is stationary')
```

```
        else:
            x = print('ADF Statistic: %f' % result[0],'\n','p-
value: %f' % result[1], '\n', yc.iloc[:,[p]].columns.get_level
_values(4)[0] + ' is not stationary')
        print(x)
    return

# Fit stepwise an auto-ARIMA or dynamic

import pmdarima as pm

sxmodel = pm.auto_arima(y1, exogenous = exog.iloc[:,[0]], star
t_p=0, start_q=0,
                        test='adf',
                        max_p=0, max_q=0,m=12,
                         start_P=0, max_P=2, start_Q=0, max_Q=
2,seasonal=True,
                        d=0, D=0, trace=True,
                        error_action='ignore',
                        suppress_warnings=True,
                        stepwise=True)

# Residual diagnosis for ARIMA and Dynamic

def bestarimaplot (model, m, n):
    model.plot_diagnostics(figsize = (m, n))
    return plt.show()
```