ERASMUS UNIVERSITY ROTTERDAM

M.SC. THESIS

Forecasting Short Life Cycle Product Demand
a Fuzzy Clustering Approach

Author:
J.M.J. de Winter B.Sc.
434523

Supervisor:
Prof. Dr. P.J.F. Groenen

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in

Econometrics & Management Science

ERASMUS UNIVERSITEIT ROTTERDAM
ERASMUS SCHOOL OF ECONOMICS

December, 2017

Abstract

This research investigates forecasting Short Life Cycle Product demand with a fuzzy clustering approach. The forecasting methods from Basallo-Triana, Rodríguez-Sarasty, and Benitez-Restrepo (2017) are verified in this research. In their framework, fuzzy clustering is used to identify clusters of analogous sales profiles. A predictive model – Multiple Linear Regression (MLR) – is then trained on each cluster. They then assign a new product to a cluster, after which it follows that specific model for prediction.

In verifying their methods, we provide proof that the use of the distance measure from Frigui and Krishnapuram (1999) does not minimise the fuzzy clustering cost function. Therefore, the original Gustafson-Kessel algorithm is implemented in this research (Gustafson & Kessel, 1979). Actual sales from a Dutch e-commerce retailer is then predicted with MLR. Forecasts are attempted to be improved with Multivariate Adaptive Regression Splines (MARS), log-transformations and a weighted forecast. The latter incorporates the fuzzy clustering approach, in which a new product is assigned to a weighted combination of all identified clusters.

In terms of Root Mean Squared Error (RMSE) of out-of-sample forecasts, it is shown that analogue-based forecasting with MLR beats a simple naive benchmark. However, a first indication on MARS show no substantial improvement compared to MLR, while it does provide a great extra computational burden. Moreover, clustering performance seems to limit model performance, resulting in poor overall forecasting performance.

Contents

Chapter 1
Introduction

Short Life-Cycle Products (SLCP) are increasingly common within the online retail industry; life cycles of only months are not remarkable within e.g. fashion and electronics industries. Accurate demand forecasts are necessary both for preventing obsolete stock and meeting high consumer demands. Moreover, forecasting insights can lead to better operational decisions in order to influence sales when desired.

1.1 Analogue Based Forecasting with Fuzzy Clustering

In recent research, Basallo-Triana et al. (2017) propose to use fuzzy clustering to identify clusters of analogous products, in the sense that they have comparable sales profiles over a short period of time. A predictive model is trained for each of these identified clusters, incorporating clustering uncertainty with fuzzy observation weights. Although not entirely in line with the fuzzy nature, they then assign a new, out-of-sample product to one cluster and predict its sales using the corresponding cluster model.

This analogue-based forecasting method is especially relevant for an e-commerce retailer, with a large assortment that consists of a wide variety of products and many different demand profiles. Using clusters of analogous sales profiles, product demand can be predicted over a cross-section of sales profiles, without needing a lot of historical data of the product itself.

The original fuzzy c-means clustering methods were developed by Dunn (1973) and later improved by Bezdek (1981). Fuzzy clustering is based on the minimisation of a cost function, which is a weighted sum over the distances between observations and the cluster means. In the original fuzzy c-means, this measure is the simple Euclidean distance, but Gustafson and Kessel (1979) transformed this measure involving *fuzzy covariance matrices*. It has been shown that this method was especially suited for the identification of hyperplanar clusters.
Basallo-Triana et al. (2017) implement the Gustafson-Kessel (GK) framework, but make use of a different distance measure as defined by Frigui and Krishnapuram (1999). This Frigui-Krishnapuram (FK) distance measure originates from Bezdek (1981) and was later modified by Davé (1990) for line clusters in 2D. Frigui and Krishnapuram (1999) generalised the distance measure to its current form.

In this research, it is shown that this distance measure is problematic when using it in the original GK framework. Mathematical arguments for this conclusion are provided by solving the fuzzy clustering minimisation with the FK distance measure. The result demonstrates that the use of the FK distance measure does not minimise the fuzzy clustering cost function. Therefore it does not result in the optimal identification of clusters. This finding is one of the main contributions of this research.

As the distance measure does not provide well-behaved clustering, the methods and results of Basallo-Triana et al. (2017) are disputable, moreover due to flawed computational implementation. Therefore, the original GK distance measure is implemented and the findings of Basallo-Triana et al. (2017) are verified. Furthermore, an alternative forecasting method is proposed to make fully use of the fuzzy nature of clustering. This method does not assume only one cluster for a new product, but uses a fuzzy weighted combination of cluster model prediction.

1.2 Predictive Models: MLR and MARS

Basallo-Triana et al. (2017) use a relatively straightforward Multiple Linear Regression (MLR) as predictive model for each cluster, including lagged sales as predictor variables. However, this model has some limitations. First, MLR cannot cope with non-linear dependencies between sales and any of the predictive variables. Secondly, the model contains no product specific information such as price (position) and moment of introduction, such that the model will not provide any operational insights to influence sales when necessary. In this research a model is investigate that could be suited to overcome these limitations. This model is called Multivariate Adaptive Regression Splines (MARS, J. H. Friedman 1991).

MARS is proven to be successful in forecasting sales and can be used to identify key operational decision variables instead of using time series data (Lu, Lee, & Lian, 2012). Opposed to MLR, MARS is able to capture non-linear dependencies between sales and descriptive variables. MARS basically divides the variable space into sub-intervals for which it performs linear regression. Hence, a continuous and possibly non-linear relation is found over the entire variable space. In order to prevent over-fitting, a second stage of the training process uses generalised cross-validation to reduce complexity of the model.

We compare the MLR and MARS models as predictive models within the fuzzy clustering framework. As results of Basallo-Triana et al. (2017) are disputable, both models are compared with a simple naive benchmark as well. MLR is shown to improve forecasting slightly compared to the naive forecast in terms of Root Mean Squared Error of predicted sales. Unfortunately, MARS does not improve forecasting performance.

1.3 Related Work

Many different predictive models have been proposed in the past to forecast SLCP demand. Besides general forecasting tools – e.g. regression approaches or Box-Jenkins models – the diffusion model from Bass (1969) tries to characterise the specific stages of SLCP demand: product introduction, maturity and decline. Various extensions of this model have been proposed by including seasonal effects (Radas & Shugan, 1998) or decision variables such as price and advertising (Bass, Krishnan, & Jain, 1994). Although showing relative success in forecasting SLCP demand, these models have limitations. Model parameters have to be tuned with data, which is especially scarce in short life cycles. Bayesian updating tries to overcome this restraint as carried out by Zhu and Thonemann (2004). However, the model still assumes life-cycle patterns that may not be observed for real products with limited sales.

Naturally, non-parametric methods have been proposed as well. These methods include the use of Artificial Neural Networks (ANN) (Zhang & Qi, 2005) and Support Vector Regressions (SVR) (Lu, 2014). These data-driven methods have clear advantages over parametric models as they do not assume any particular sales profiles and are able to capture non-linear effects. However, these models are prone to over-fitting, models often lack clear interpretation and performance seems strongly dependent on the specific application and often results.

Other methods using clustering exist as well and also try to identify clusters of demand profiles within historical sales, e.g. the method of Thomassey and Happiette (2007). In this cold start approach, no time series data of new products is necessary for prediction, as classification can be done by e.g. price, expected lifespan and starting date. Although this model showed relative success, the research was conducted on specific industries and generalisation is not easily carried out.

1.4 Research Scope & Structure

The main objective of this research is to verify whether SLCP demand can be predicted by means of analogue-based forecasting with fuzzy clustering. To this extent, Chapter 3 provides a detailed description of the fuzzy clustering algorithm. In this chapter it is also proven mathematically why the Frigui-Krishnapuram distance measure does not minimise the fuzzy clustering cost function.

Furthermore, it is investigated whether MLR can be used as predictive model for the identified clusters, and whether it can be improved by using MARS. Both models are defined in Chapter 4 in respect to this research and the actual one-step-ahead forecasting procedure is described in this chapter as well. Chapter 5 describes the evaluation procedure and produced results. The research is concluded with recommendations in Chapter 6.

All empirical results in this research are based on actual sales and offer data from a Dutch e-commerce retailer. To this extent, the following Chapter 2 describes the data acquisition, clean-up procedure and its characteristics.

Chapter 2
Data

The data used for this research are based on actual offer and sales data from a Dutch e-commerce retailer. The data are retrieved for products within the electronics and entertainment sectors. Products in the data set range from video games to computers and other hardware, mimicking the actual product assortment. For products with sales starting after 1 January 2016 daily data is retrieved up to 31 August 2017 - a period of around 80 weeks.

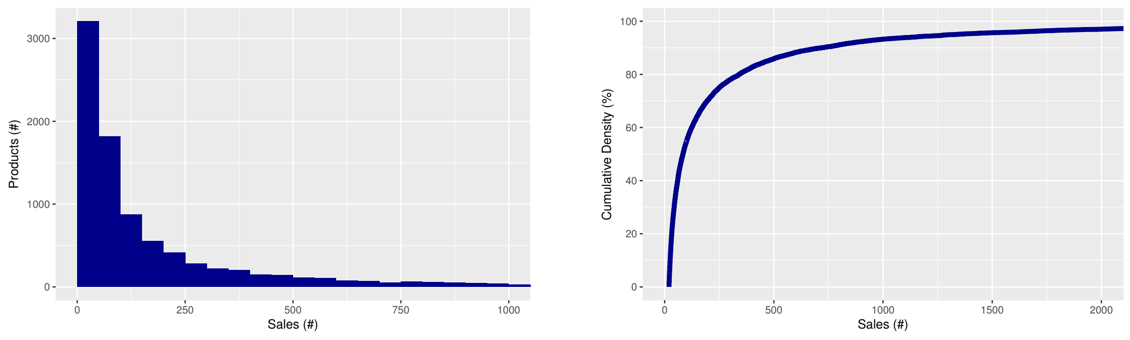
The data consists of time series of 9149 products and combines daily data for the following observable variables:

- Quantity of daily sales;
- Consumer price (€) of the product. If sales data is available, the daily average of the price for which the goods are bought is used. For days without any sales a snapshot of the price is taken at the end of each day at 24:00 hours;
- Price position of the product relative to market. This information is captured in an integer ranging from 1 - very high price compared to market - to 5 (very low price compared to market). This value is calculated with benchmark information from other sellers or e.g. consumer advice price (CAP). If no information is available, a value of zero is assigned to the specific offer;
- Time information, retrieved as dummy variable for the week number;
- Orderability of the product, as a product can be taken offline and hence not be sold;
- Delivery information, captured in a dummy variable on the minimal delivery time, ranging from 1 to 5.

As the data is prone to manual intervention it is necessary to clean-up data by e.g. filling in gaps of data. Also, to be able to use the data for clustering, normalisation is carried out. This process is carried out in the following steps:

- Missing days of data are filled in by substituting previous day data, given the fact that products were actually online and orderable;
- The first day of sales for every product is normalised so that all sales profiles start on the same fictive time unit. Of course, dummy information on time remains;
- Data is aggregated over the longer interval of 7 days, in order to smoothen sales data. Sales data is summed, the minimum price and delivery information are stored and the maximum value for price position is used. Aslo, the percentage of the interval for which the product was online is also used, e.g. $\sim$14% if a product was only orderable for only 1 out of 7 days;
- Only products are taken into account with total sales over the sample period between 20 and 2000 units. With a very large assortment, the majority of the products have few sales, as can be seen in Figure 2.1. Products with too few sales are taken out, because this would allow for many zero sales profiles within the data, reducing computation speed greatly. Products with high sales are taken out as well, to prevent too many outliers within the data. Besides, these products are generally monitored often and an expert view can cope with the forecasts. Figure 2.1 shows the distribution of products over the sales within

the sample period. Especially from Figure 2.1b it can be observed that most density lies within products with few sales. For example, by only taking into account products with 500 sales or less, over 80% of the data set is covered.



(A) Histogram (B) Cumulative Density

FIGURE 2.1: Distribution info of the data set with respect to total sales. Figure (a) contains a histogram showing how many products have a specific range of total sales. Figure (b) shows a cumulative density, indicating how many products (%) have total sales equal or less than a specfic amount.

2.1 Autocorrelation in Sales

Basallo-Triana et al. (2017) use lagged sales as predictor variables in MLR. They show substantial autocorrelation in their data, which provides an extra argument to use lagged sales in their model.

For the data set, highly fluctuating autocorrelation is observed for the various products within the selection. Sample autocorrelation for two illustrative products is given in Table 2.1, demonstrating fluctuating autocorrelations between two products with high sales. This heterogeneity in autocorrelation across the product selection can make it cumbersome to provide a single predictive model for the entire product selection. This provides extra support for clustering of sales profiles and modelling each cluster independently; differences in explanatory power of the variables need different models to forecast sales accurately. A second observation is that autocorrelation can be very low, which is even worse for products with very few sales. This indicates it can be useful to look into other explanatory variables besides lagged sales.

TABLE 2.1: For two illustrative products, the sample auto-correlation within four-day interval sales is shown. Up to five lags are taken into account.

Product	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5
Fantastic Beasts and Where to Find Them	0.84	0.72	0.59	0.51	0.41
Sony Action-Cam HDR-AS50	0.29	0.28	0.17	0.11	0.22

Chapter 3
Fuzzy Clustering

This section provides a detailed description of the fuzzy clustering algorithm as used in this research. In this approach, the methods of Basallo-Triana et al. (2017) are not followed. In their clustering effort, they make use of the FK distance measure defined by Frigui and Krishnapuram (1999). In this chapter, mathematical proof is given that this distance measure does not minimise the fuzzy clustering cost function, and thus the methods of Basallo-Triana et al. (2017) are incorrect. Before describing this result, the general fuzzy clustering problem definition is provided. This problem is then solved for the Gustafson-Kessel (GK) distance measure (Gustafson & Kessel, 1979), which is used throughout this research.

3.1 General Framework

Fuzzy clustering is used to find clusters of analogous sales profiles for each time t. Predictive models can then be trained to suit the specific cluster of analogies. Opposed to hard clustering, an observation can belong to multiple clusters simultaneously and thus can mitigate bias imposed by hard clustering. The general framework of Bezdek, Ehrlich, and Full (1984) is demonstrated, including the distance measure of Gustafson and Kessel (1979). Notation and computational implementation are based on the work of Theodoridis and Koutroumbas (2008).

On each time instance t, the data consists of N observation vectors $\mathbf{x}_i$. Each vector contains a short sales profile based on $\ell = p + 1$ time periods, with p indicating the number of lags included. This results in observation vectors $\mathbf{x}_i = (y_{i,t-p}, \dots, y_{i,t})$, where $y_{i,t}$ denotes the sales of product i on time t. The observations, collected in the $N \times \ell$ matrix $\mathbf{X}$, are clustered towards K clusters. In fuzzy clustering, any observation vector $\mathbf{x}_i$ $(i = 1 \dots N)$ belongs to some extent to any of the clusters $k = 1 \dots K$, which is captured by the grade of membership $u_{ik} \in [0, 1]$. In general, a cluster is characterised by a point representation (cluster centre or centroid) $\mathbf{c}_k$, which has the same dimensions as the observation vectors.

The elements u_{ik} of the $N \times K$ matrix $\mathbf{U}$, and values of $\mathbf{c}_k$ of the $K \times \ell$ matrix $\mathbf{C}$ are found by minimising the following cost function with respect to all its variables (Bezdek et al., 1984; Theodoridis & Koutroumbas, 2008):

$$\begin{aligned} J_q(\mathbf{U}, \mathbf{C}) &= \sum_{i=1}^{N} \sum_{k=1}^{K} u_{ik}^q \, d^2(\mathbf{x}_i, \mathbf{c}_k), \\ s.t. \quad \sum_{k=1}^{K} u_{ik} &= 1, \qquad u_{ik} \geq 0. \end{aligned} \tag{3.1}$$

For this cost function, one must specify the value of $q(\geq 1)$, which is called the *fuzzifier*. This fuzzifier impact the extent to which the clustering is fuzzy. In the limit $q \to 1$, the values of u_{ik} will converge to either 0 or 1, implying hard or crisp clustering. For the limit $q \to \infty$, no clustering takes place, i.e. all observations will belong to one cluster. For practical use this value is often set to $q = 2$ in absence of any hard evidence against this choice. In this research ths value is used as well.

The cost function of (3.1) uses $d_{ik}^2 = d^2(\mathbf{x}_i, \mathbf{c}_k)$, which is a distance measure between any observation vector and any of the cluster centroids. All values of d_{ik}^2 are contained in the $N \times K$ matrix $\mathbf{D}$. The choice of distance measure naturally influences the solution to the minimisation. A most simple form of this distance measure would be the use of the Euclidean distance, resulting in spherical clusters.

First the solution of this mathematical problem is provided for the Gustafson-Kessel (GK) distance measure. It has been shown that this distance measure performs well for hyperplanar clusters and is often used in e.g. image line reconstructions. In a similar fashion, it is then argued why the methods of Basallo-Triana et al. (2017) are incorrect, since it is proven mathematically that the use of their FK distance measure (Frigui & Krishnapuram, 1999) does not minimise (3.1).

3.2 Gustafson-Kessel Distance Measure and Solution

Gustafson and Kessel (1979) use the following general distance measure:

$$d_{GK}^2(\mathbf{x}_i, \mathbf{A}_k, \mathbf{c}_k) = (\mathbf{x}_i - \mathbf{c}_k)'\mathbf{A}_k(\mathbf{x}_i - \mathbf{c}_k), \tag{3.2}$$

in which $\mathbf{A}_k$ is a symmetrical and positive definite matrix of dimensions $\ell \times \ell$, similar to the length of the vectors $\mathbf{x}_i$ and $\mathbf{c}_k$. As the cost function from (3.1) is linear in $\mathbf{A}_k$, minimising with respect to $\mathbf{A}_k$ is meaningless, as it would result in $\mathbf{A_k} = \mathbf{0}$. Gustafson and Kessel (1979) solve this problem by adding an extra constraint:

$$|\mathbf{A}_k| = \rho_k, \qquad \rho_k > 0, \tag{3.3}$$

which implies a volume constraint. Gustafson and Kessel (1979) propose to set $\rho_k = 1$, which is also used in this research.

The values for $\mathbf{U}$, $\mathbf{c}_k$ and $\mathbf{A}_k$ are found by minimising a Lagrangian based on (3.1) and the constraint from (3.3). This derivation can be found in Appendix A and results in the following solution of u_{ik}, $\mathbf{c}_k$ and $\mathbf{A}_k$:

$$u_{ik} = \frac{d^2(\mathbf{x}_i, \mathbf{c}_k)^{-1/q-1}}{\sum_{l=1}^{K} d^2(\mathbf{x}_i, \mathbf{c}_l)^{-1/q-1}} \tag{3.4a}$$

$$\mathbf{c}_k = \frac{\sum_{i=1}^{N} u_{ik}^q \mathbf{x}_i}{\sum_{i=1}^{N} u_{ik}^q} \tag{3.4b}$$

$$\mathbf{A}_k = [\rho_k |\mathbf{F}_k|]^{1/\ell} \, \mathbf{F}_k^{-1}, \tag{3.4c}$$

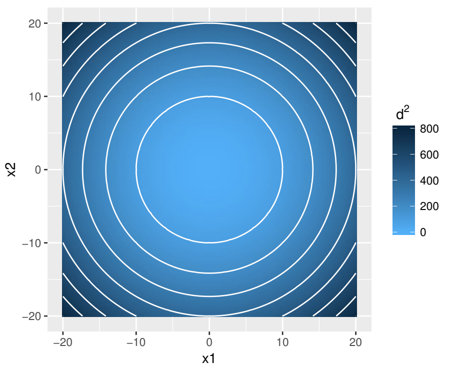
in which the *fuzzy covariance matrix* $\mathbf{F}_k$ is defined as

$$\mathbf{F}_k = \frac{\sum_{i=1}^{N} u_{ik}^q (\mathbf{x}_i - \mathbf{c}_k)(\mathbf{x}_i - \mathbf{c}_k)'}{\sum_{i=1}^{N} u_{ik}^q}. \tag{3.4d}$$

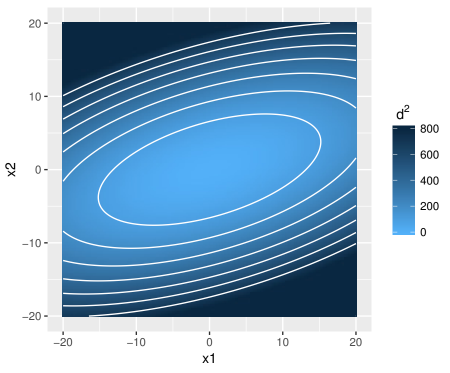
With this result, every cluster is characterised by its centroid $\mathbf{c}_k$ and a weighted covariance matrix $\mathbf{F}_k$. It can be observed that the distance measure is normalised by this fuzzy covariance matrix, allowing clusters to vary in shape. This behaviour can be observed in Figure 3.1, in which a heat map and contour plot of two different distance measures $d^2(\mathbf{x}_i, \mathbf{0})$ is plotted. These plots show the value of the distance measure for two possible 2D clusters with centroids at $(0, 0)$. In Figure 3.1a a fuzzy covariance matrix $\mathbf{F}_A = (1\ 0; 0\ 1)$ is used for the cluster, while

Figure 3.1b shows the distance measure of a cluster with $\mathbf{F}_B = (0.25\ 0.0625; 0.0625\ 0.0625)$.

With equations (3.2) and (3.4c) the resulting transformed distance measures is calculated. It can be observed that unity matrix $\mathbf{F}_A$ results in the Euclidean distance. However, $\mathbf{F}_B$ results in a transformed distance measure, such that observations are penalised less if they are close to the shape and direction of the cluster covariance matrix. Therefore, the inclusion of the fuzzy cluster covariance matrix allows for variations in shape and direction of the clusters.



(A) Euclidean Distance



(B) Transformed Distance

FIGURE 3.1: Heat map and contour plot of distance measure from (3.2) for 2D examples with cluster centres at $(0, 0)$, calculated for a 2D grid of possible observation vectors. Two matrices $\mathbf{A}_k$ are used: (a) unity matrix for Euclidean distance and (b) transformed matrix.

3.3 Computational Implementation

Since the updates in (3.4) are coupled, an iterative algorithm is necessary for their solutions. Algorithm 1 shows the GK algorithm, an adaptation of the Generalised Fuzzy Algorithmic Scheme (GFAS) from Theodoridis and Koutroumbas (2008) that is used to solve for the parameters numerically.

This algorithm can be computationally challenging, especially in computing the determinant and inverse of the covariance matrix in (3.4c). $\mathbf{F}_k$ might not be positive definite, i.e. $|\mathbf{F}_k| \leq 0$. If this is the case,the matrix cannot be inverted and taking the ℓ-th root of the determinant is not possible. One of the possible causes for this problem is that there are few observations within a cluster or that they are co-linear, causing the rank of the matrix to be smaller than its dimensions.

We follow Babuska, Van der Veen, and Kaymak (2002) to improve covariance estimation in (3.4d). They propose adding a scaled identity matrix to the fuzzy covariance matrix in each iteration of Algorithm 1:

$$\begin{aligned} \mathbf{F}_k^{new} &= \gamma \mathbf{F}_k + (1-\gamma)|\mathbf{F}_0|^{1/\ell}\mathbf{I}_\ell, \\ \mathbf{F}_0 &= \mathrm{cov}(\mathbf{X}), \end{aligned} \tag{3.5}$$

as well as bounding the eigenvalues of the new matrix $\mathbf{F}_k^{new} = \mathbf{E}_k \mathbf{\Lambda} \mathbf{E}_k'$:

$$\begin{aligned} \lambda_{k,l} &= \lambda_{k,\max}/\beta \qquad \forall l \text{ for which } \lambda_{k,\max}/\lambda_{k,l} > \beta, \\ \lambda_{k,\max} &= \max_{l=1\dots\ell} \lambda_{k,l}. \end{aligned} \tag{3.6}$$

The necessary tuning parameters β and γ are set to 10^{15} and 10^{-5} respectively. In practice, these adjustments cause the covariance matrix to be positive definite. Babuska et al. (2002) show the altered algorithm still converges monotonically, with only minor accuracy losses. It should be noted that another possible solution for the inversion of singular matrices would be to use the Moore-Penrose pseudo-inverse. However, while this allows one to invert $\mathbf{F}_k$, it is observed that the algorithm does not converges monotonically. An example of this behaviour can be observed in Appendix B. Hence, this solution is not pursued in this research.

A second drawback of the fuzzy clustering algorithm is convergence to a local minimum. Bezdek and Hathaway (1992) show that these type of algorithms do converge to a local minimum, but the first guess of **U** can alter the local minimum to which the algorithm converges. To overcome this problem, random initialisation of **U** can be repeated, which may provide more confidence of having found an optimal solution of the fuzzy clustering.

Algorithm 1 Gustafson-Kessel (GK) algorithm for fuzzy clustering

```
Set $\tau \leftarrow 0$.

Choose an initial random estimate for $\mathbf{U}^\tau$, such that the constraint of (3.1) holds.

With the values of $\mathbf{U}^\tau$, calculate $\mathbf{D}^\tau$, $\mathbf{C}^\tau$, $\mathbf{A}_k^\tau$ and $\mathbf{F}_k^\tau$ using (3.2) and (3.4b) - (3.4d) respectively.

while $\max_{i,k} \left| u_{ik}^\tau - u_{ik}^{\tau-1} \right| < \varepsilon$ do

    Set $\tau \leftarrow \tau + 1$

    Update grades of membership:
    Calculate $\mathbf{U}^\tau$ using (3.4a) with $\mathbf{D}^{\tau-1}$.

    Calculate parameters:
    Calculate $\mathbf{C}^\tau$ using (3.4b) with $\mathbf{U}^\tau$.
    Calculate $\mathbf{F}_k^\tau$ using (3.4d). This matrix is updated for robustness using (3.5, 3.6)
    Calculate $\mathbf{A}_k^\tau$ using (3.4c) with $\mathbf{F}_k^\tau$.
    Calculate $\mathbf{D}^\tau$ using (3.2) with $\mathbf{U}^\tau$, $\mathbf{C}^\tau$ and $\mathbf{A}_k^\tau$.

end while
```

3.4 Improper Frigui-Krishnapuram Distance Measure

Basallo-Triana et al. (2017) do not use the GK distance measure from (3.2). They use the FK distance measure which was originally defined by Bezdek (1981). Davé (1990) modified the definition for 2D clusters and Frigui and Krishnapuram (1999) finally generalised this distance measure to more dimensions. This distance measure uses a decomposition of the fuzzy covariance matrix $\mathbf{F}_k = \mathbf{E}_k \mathbf{\Lambda}_k \mathbf{E}_k'$ from (3.4d). The matrix $\mathbf{\Lambda}_k$ contains the eigenvalues of $\mathbf{F}_k$ on its diagonals in descending order. The eigenvector matrix $\mathbf{E}_k$ is ordered correspondingly as follows:

$$\mathbf{E}_k = \begin{pmatrix}\mathbf{e}_{k,1} & \dots & \mathbf{e}_{k,\ell}\end{pmatrix}, \qquad \mathbf{\Lambda}_k = \begin{pmatrix}\lambda_{k,1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_{k,\ell}\end{pmatrix}. \tag{3.7}$$

The Frigui-Krishnapuram (FK) distance measure is then defined as follows:

$$d^2_{FK}(\mathbf{x}_i, \mathbf{c}_k) = \lambda_{k,1}(\mathbf{x}_i - \mathbf{c}_k)'\mathbf{E}_k\mathbf{\Lambda}_k^{-1}\mathbf{E}_k'(\mathbf{x}_i - \mathbf{c}_k). \tag{3.8}$$

The distance measure is quite similar to the GK distance measure, but it is scaled by the largest eigenvalue of $\mathbf{F}_k$. With this FK distance measure Frigui and Krishnapuram (1999) – and Basallo-Triana et al. (2017) all the same – use (3.4a), (3.4b) and (3.4d) as solution to the minimisation problem, exactly similar to Gustafson and Kessel (1979). Strictly speaking however, the original solution as presented above does not minimise the cost function from (3.1) when using the distance measure of Frigui and Krishnapuram (1999).

3.4.1 Framework of Minimisation with FK Distance Measure

In order to proof the FK distance measure cannot be used to minimise (3.1) and hence the methods of Basallo-Triana et al. (2017) are erroneous, an exact solution to the fuzzy clustering minimisation of (3.1) is provided with a distance measure in the form of (3.8). To this extent (3.8) is generalised:

$$d^2_G(\mathbf{x}_i, \mathbf{c}_k) = (\mathbf{x}_i - \mathbf{c}_k)'\mathbf{E}_k\mathbf{\Gamma}_k^{-1}\mathbf{E}_k'(\mathbf{x}_i - \mathbf{c}_k). \tag{3.9}$$

This general form is matched to the FK distance measure, in the sense that similar constraints are assumed on the form and values of $\mathbf{E}_k$ and $\mathbf{\Gamma}_k$. $\mathbf{E}_k$ is therefore defined as a general $\ell \times \ell$ matrix, but with the constraint that $\mathbf{E}_k'\mathbf{E}_k = \mathbf{I}_\ell$. The diagonal $\ell \times \ell$ matrix $\mathbf{\Gamma}_k$ is defined as:

$$\mathbf{\Gamma}_k = \begin{pmatrix}\gamma_{k,1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \gamma_{k,\ell}\end{pmatrix}, \qquad \begin{cases}\gamma_{k,l} = 1, & l = 1 \\ 0 < \gamma_{k,l} \leq 1 & l = 2 \dots \ell\end{cases} \tag{3.10}$$

With the extra constraints, d^2_G attains a similar range of possible values as the FK distance measure. If the FK distance measure were to be correct, the values of $\mathbf{E}_k$ and $\mathbf{\Gamma}_k$ that minimise (3.1) should result in values such that $d^2_G = d^2_{FK}$. The values of $\mathbf{U}$ and $\mathbf{C}$ are not explicitly calculated here. Following the derivation of Appendix A, it can be easily shown that both d^2_{FK} and d^2_G result in the same solutions (3.4a) and (3.4b) for these values.

For the solution of $\mathbf{E}_k$ and $\mathbf{\Gamma}_k$, d^2_G is substituted into (3.1). Since only a minimisation with respect to $\mathbf{E}_k$ and $\mathbf{\Gamma}_k$ is of interest, only relevant terms are collected, assuming all other terms remain constant. This results in the following altered cost function:

$$J(\mathbf{E}_k, \mathbf{\Gamma}_k) = \sum_{i=1}^{N} u_{ik}^q(\mathbf{x}_i - \mathbf{c}_k)'\mathbf{E}_k\mathbf{\Gamma}_k^{-1}\mathbf{E}_k'(\mathbf{x}_i - \mathbf{c}_k). \tag{3.11}$$

3.4.2 Derivation of Minimisation

To solve the minimisation with respect to $\mathbf{E}_k$, first (3.11) is rewritten. Since the resulting value of the cost function in (3.11) is a scalar, a trace operator can be applied without changing its value. This trace operator allows rearranging the terms cyclically (Petersen & Pedersen, 2012):

$$\begin{aligned} J(\mathbf{E}_k, \mathbf{\Gamma}_k) &= \mathrm{Tr}\left[\sum_{i=1}^{N} u_{ik}^q (\mathbf{x}_i - \mathbf{c}_k)'\mathbf{E}_k\mathbf{\Gamma}_k^{-1}\mathbf{E}_k'(\mathbf{x}_i - \mathbf{c}_k)\right] \\ &= \mathrm{Tr}\left[\mathbf{E}_k' \sum_{i=1}^{N} u_{ik}^q (\mathbf{x}_i - \mathbf{c}_k)(\mathbf{x}_i - \mathbf{c}_k)'\mathbf{E}_k\mathbf{\Gamma}_k^{-1}\right] \\ &= \mathrm{Tr}\left[\mathbf{E}_k'\mathbf{G}_k\mathbf{E}_k\mathbf{\Gamma}_k^{-1}\right]. \end{aligned} \tag{3.12}$$

Here, $\mathbf{G}_k = \sum_{i=1}^{N} u_{ik}^q (\mathbf{x}_i - \mathbf{c}_k)(\mathbf{x}_i - \mathbf{c}_k)'$, similar to Appendix A.

Applying the trace operator to the cost function allows us to use *Kristof's upper bound* for trace functions (Kristof, 1969). Following this theorem, it is possible to find the values of $\mathbf{E}_k$, without the need of solving a Lagrangian.

The theorem of Kristof (1969) and its generalisation by Ten Berge (1983) states the following. Define $\mathbf{C}_i$ to be $\ell \times \ell$ orthonormal matrices and $\mathbf{D}_i$ to be $\ell \times \ell$ diagonal matrices, with its diagonal elements in weakly descending order and $i = 1, \ldots, n$. It is proven that limits can be attained for

$$\mathrm{Tr}\left[\mathbf{C}_1\mathbf{D}_i \cdots \mathbf{C}_n\mathbf{D}_n\right] \leq \mathrm{Tr}\left[\mathbf{D}_1 \cdots \mathbf{D}_n\right]. \tag{3.13}$$

This theorem is not directly applicable to (3.12) for two reasons. First, the theorem finds an upper bound for trace functions while in this case a minimisation is required. Hence, the sign of the trace function is changed and maximisation can be performed. Secondly, $\mathbf{G}_k$ is not diagonal in structure. However, it is assumed an eigenvalue decomposition of $\mathbf{G}_k$ exists. Since $\mathbf{G}_k$ is symmetric by definition this results in $\mathbf{G}_k = \mathbf{K}_k\mathbf{\Theta}_k\mathbf{K}_k'$. The column vectors of $\mathbf{K}_k$ and diagonal elements of $\mathbf{\Theta}_k$ are arranged in descending order of the eigenvalue. Multiplying (3.12) by minus one and substituting the eigenvalue decomposition results in:

$$\tilde{J}(\mathbf{E}_k, \mathbf{\Gamma}_k) = \mathrm{Tr}\left[-\mathbf{E}_k'\mathbf{K}_k\mathbf{\Theta}_k\mathbf{K}_k'\mathbf{E}_k\mathbf{\Gamma}_k^{-1}\right] \tag{3.14}$$

From their definitions, both $\mathbf{E}_k$ and $\mathbf{K}_k$ are orthonormal and the product of the two matrices is also orthonormal (Ten Berge, 1983). Now it is possible to use Kristof's theorem and the limits for

$$\mathrm{Tr}\left[-\mathbf{E}_k'\mathbf{K}_k\mathbf{\Theta}_k\mathbf{K}_k'\mathbf{E}_k\mathbf{\Gamma}_k^{-1}\right] \geq \mathrm{Tr}\left[-\mathbf{\Theta}_k\mathbf{\Gamma}_k^{-1}\right] \tag{3.15}$$

are attained if both $\mathbf{E}_k'\mathbf{K}_k$ and $\mathbf{K}_k'\mathbf{E}_k$ are equal to $\mathbf{I}_\ell$. From its definition it is known that $\mathbf{K}_k'\mathbf{K}_k = \mathbf{I}_\ell$. Hence, limits of this trace function are attained if $\mathbf{E}_k = \mathbf{K}_k$.

This implies that $\mathbf{E}_k$ can be retrieved with the eigenvalue decomposition of $\mathbf{G}_k$. It can be easily verified that this is the same as retrieving the eigenvalue decomposition of the fuzzy covariance matrix $\mathbf{F}_k$, defined in (3.4d). Hence, the matrix $\mathbf{E}_k$ that minimises the fuzzy clustering cost function is found by the eigenvalue decomposition of $\mathbf{F}_k = \mathbf{E}_k\mathbf{\Lambda}_k\mathbf{E}_k'$. This is actually similar to the FK distance measure.

However, the solution of $\mathbf{\Gamma}_k$ is different from Frigui and Krishnapuram (1999). From (3.11) and the diagonal structure of $\mathbf{\Gamma}_k$, it can be observed that the cost function is linear in each diagonal element $\gamma_{k,l}^{-1}$. This implies (3.11) is minimised if all individual $\gamma_{k,l}^{-1}$ are at their minimum. The constraints from (3.10) show that each element $\gamma_{k,l}$ is bounded between 0 and 1, which

imply that the range of each $\gamma_{k,l}^{-1}$ is given by $1 \leq \gamma_{k,l}^{-1} < \infty$. The first element $\gamma_{k,1}$ is just a special case of this argument and its value is already fixed at 1.
Hence, it is observed that $\mathbf{\Gamma}_k = \mathbf{I}_\ell$ minimises the fuzzy clustering cost function.

3.4.3 Solution to FK Distance Measure

Substituting the solutions for $\mathbf{\Gamma}_k$ and $\mathbf{E}_k$ into (3.9) it is observed that the distance measure reduces to:

$$\begin{aligned} d_G^2(\mathbf{x}_i, \mathbf{c}_k) &= (\mathbf{x}_i - \mathbf{c}_k)'\mathbf{E}_k\mathbf{\Gamma}_k^{-1}\mathbf{E}_k'(\mathbf{x}_i - \mathbf{c}_k) \\ &= (\mathbf{x}_i - \mathbf{c}_k)'\mathbf{E}_k\mathbf{I}_\ell\mathbf{E}_k'(\mathbf{x}_i - \mathbf{c}_k) \\ &= (\mathbf{x}_i - \mathbf{c}_k)'(\mathbf{x}_i - \mathbf{c}_k). \end{aligned} \tag{3.16}$$

This is the regular Euclidean distance between an observation and a cluster centroid. Since this is definitely not equal to (3.8), it is proven that the FK distance measure does not minimise the fuzzy clustering cost function. In practice this can e.g. lead to a non decreasing and non converging cost function (3.1) over the iterations of the algorithm.

In this research, the FK distance measure is not used in the clustering algorithm like Basallo-Triana et al. (2017). Instead, the original GK algorithm is implemented.

3.4.4 Alternative Distance Measure as Adaptive Fuzzy Clustering

Although it is proven that the methods of Frigui and Krishnapuram (1999) are problematic, they provide some extra intuition and reasoning as to why they use the FK distance measure. To this extent, they compare the GK and FK distance measures, since both measures should be equal if they result in the same solution.

Substituting the definition of $\mathbf{A}_k$ from (3.4c) into the general distance measure of (3.2) results in:

$$d_{GK}^2(\mathbf{x}_i, \mathbf{c}_k) = \left[\rho_k|\mathbf{F}_k|\right]^{1/\ell}(\mathbf{x}_i - \mathbf{c}_k)'\mathbf{F}_k^{-1}(\mathbf{x}_i - \mathbf{c}_i). \tag{3.17}$$

This expression is comparable to (3.8), by using the eigenvalue decomposition $\mathbf{F}_k^{-1} = \mathbf{E}_k\mathbf{\Lambda}_k^{-1}\mathbf{E}_k'$. If both distance measures result in the same solution, this imposes:

$$\rho_k = \frac{\lambda_{k,1}^\ell}{|\mathbf{F}_k|} \tag{3.18}$$

Strictly this is not a valid relation for ρ_k, since the problem as defined by Gustafson and Kessel (1979) require a predefined, fixed value for ρ_k in (3.3). For the distance measure d_{FK}^2, however, this value is revised in every iteration of Algorithm 1, hoping that it improves the initial estimate of ρ_k. Hence, the FK distance measure can be seen as an adaptive clustering algorithm. Although its experimental performance is demonstrated in several papers, e.g. the paper of (Davé, 1990), it is proven here that this distance measure does not minimise the fuzzy clustering cost function. Hence, the original GK distance measure is used in the clustering effort of this research.

Chapter 4
Forecasting Models

Clusters of historical sales profiles can be identified using Algorithm 1 on each time instance $t-1$. For each of these clusters, a predictive model is trained. To forecast the sales of a new product at time t, denoted by $\hat{y}_{i,t}$, this product is first assigned to one of the clusters using a minimum distance criteria. Its sales can then be predicted using the trained model of the respective cluster.

In this research, two predictive models are investigated: Multiple Linear Regression (MLR) and Multivariate Adaptive Regression Splines (MARS). This section first provides a description of both models. Second, the forecasting procedure is described to predict sales of new products using historical sales.

4.1 Multiple Linear Regression

Basallo-Triana et al. (2017) suggested to use fairly straightforward Multiple Linear Regression (MLR) to forecast sales y_t, using p lagged sales as predictive variables. This choice is backed up by significant autocovariance within the sales series used in their research. This autoregressive regression can be considered cross-sectional and is given as follows:

$$\mathbf{y}_t = \mathbf{B}_t \mathbf{w}_t + \epsilon_t, \tag{4.1}$$

where $\mathbf{y}_t$ contains the sales of all products at time t, $\mathbf{w}$ is the parameter vector and $\mathbf{B}_t$ contains the observation vectors of lagged sales preceded by ones to incorporate an intercept. These matrices are thus build up as follows:

$$\mathbf{w}_t = \begin{pmatrix} w_0^t \\ \vdots \\ w_p^t \end{pmatrix}, \qquad \mathbf{B}_t = \begin{pmatrix} 1 & y_{1,t-1} & \cdots & y_{1,t-p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & y_{N,t-1} & \cdots & y_{N,t-p} \end{pmatrix}. \tag{4.2}$$

It can be easily verified that $\mathbf{y}_t$ and $\mathbf{B}_t$ are build up from the matrix $\mathbf{X}_t$, which is the observation matrix from Section 3. Such a linear regression model is solved by minimising the sum of squared errors $(\mathbf{y_t} - \mathbf{B}_t\mathbf{w}_t)' (\mathbf{y}_t - \mathbf{B}_t\mathbf{w}_t)$ with respect to $\mathbf{w_t}$.

Equation (4.1) would hold if this model is trained on all products with equal weights. However, this model is trained for each cluster, using grades of membership as observation weights. This means the weighted sum of squared errors is minimised for every clustermodel k:

$$\min_{\mathbf{w_t^k}} \left(\mathbf{y_t} - \mathbf{B}_t\mathbf{w}_t^k\right)' \ \mathrm{diag}(\mathbf{u}_t^k) \ \left(\mathbf{y_t} - \mathbf{B}_t\mathbf{w}_t^k\right), \tag{4.3}$$

where $\mathrm{diag}(\mathbf{u}_t^k)$ is the diagonalization of column k of $\mathbf{U}_t$. This minimisation results in the weighted least squares estimator for $\mathbf{w_t^k}$ (J. Friedman et al., 2001)

$$\hat{\mathbf{w}}_t^k = \left(\mathbf{B}_t' \ \mathrm{diag}(\mathbf{u}_t^k) \ \mathbf{B}_t\right)^{-1} \mathbf{B}_t' \ \mathrm{diag}(\mathbf{u}_t^k) \ \mathbf{y}_t, \tag{4.4}$$

In this weighted multiple linear regression, the grades of membership are used that each observation has towards a cluster k. This info is contained in the weight matrix $\mathbf{U}_t^k = \text{diag}(u_{1k}, \ldots, u_{Nk})$. It should be noted that this weighted least squares can also be accomplished by multiplying both the sales vector $\mathbf{y_t}$ and observation matrix $\mathbf{B}_t$ by $\mathbf{U}_t^{k^{1/2}}$, with $\mathbf{U}_t^k = (\mathbf{U}_t^{k^{1/2}})'(\mathbf{U}_t^{k^{1/2}})$, and performing a standard linear regression. It can be easily verified that this yields the same result as equation (4.4).

In the MLR model of Basallo-Triana et al. (2017) only lagged sales are used as explanatory variable. However, it surely is possible to perform MLR with other predictive variables as well (e.g. price position). Both the parameter vector $\mathbf{w}_t^k$ and observation matrix $\mathbf{B}_t$ are then extended to incorporate these variables.

4.2 MARS

MLR is only capable of handling linear relationships between sales and any of its predictive variables. MARS, however, can be used to approximate non-linear dependencies by means of piece-wise linear regressions. Following J. Friedman et al. (2001), the general MARS function of sales y as function of variables B is given by:

$$\begin{aligned} y(B) &= w_0 + \sum_{s=1}^{S} w_s h_s(B), \\ h_0(B) &= 1. \end{aligned} \tag{4.5}$$

In this model, $h_s(B)$ is a function from $\mathcal{C}$, which is a set of of piece-wise linear basis functions that can be considered as linear *splines* and are denoted here as reflected pairs:

$$\mathcal{C} = \left\{ (B_j - b_{ij})_+, (b_{ij} - B_j)_+ \right\}, \qquad {}_{\substack{i=1,\ldots,N \\ j=1,\ldots,J}}. \tag{4.6}$$

In this function, b_{ij} denotes the location of the knot, which is one of the observations in the matrix of predictors $\mathbf{B}_t$. B_j denotes the parameter space of one variable j from the entire parameter space B. The set $\mathcal{C}$ contains up to $2NJ$ basis functions; from this set, $h_s(B)$ can be any of the basis functions or a product of two or more of these functions. Given the choice for all functions $h_s(B)$ in equation 4.5, the corresponding coefficients w_s are estimated by minimising the residual sum of squared errors (RSS).

In this effort, it is also possible to incorporate observation weights like in the MLR model. As explained with (4.4), observation weights can be incorporated by multiplying observation and sales vector with the diagonal weight matrix $(\mathbf{U}_l^k)^{1/2}$. Since MARS essentially minimises RSS, this approach still holds and although used in this research, it is not written down for clarity.

The MARS procedure is all about optimising the construction of $h_s(B)$ from basis functions from $\mathcal{C}$. This is done in two stages:

1. **Forward Building**
   Starting from the first basis function $h_0(B) = 1$, each iteration adds a reflected pair from the set $\mathcal{C}$ to the model. All possible products of functions $h_s()$ - existing in the model already - with one of these reflected pairs are considered, resulting in a term of the form:

$$\hat{w}_1 h_s(B) \cdot (B_j - b_{ij})_+ + \hat{w}_2 h_s(B) \cdot (b_{ij} - B_j)_+. \tag{4.7}$$

The weightings w_s from equation (4.5) are estimated by least squares. Only one term is actually added to the model, which is the one that results in the greatest decrease in RSS. The forward building procedure is carried out until a preset maximum number of terms is reached or the relative change in RSS becomes smaller than a preset value.

Two restrictions are imposed in this stage of the MARS procedure. First, no term is allowed to be multiplied by itself, preventing higher order *splines*. Second, the model can be made additive by excluding multiplications of basis functions. These restrictions improve speed and stability, since a lot of possible combinations do not have to be examined.

2. **Backward Deletion**
The first stage typically overfits the data and a backward removal procedure is necessary to decrease the complexity of the model. This decrease in complexity makes the model more generalised, and probably better suitable for out-of-sample forecasts. With every step in this procedure, the term that increases the RSS least is removed, resulting in an optimal model $\hat{y}_\alpha$ with α terms in it. Although RSS increases, the complexity of the model decreases. The following 'generalised cross validation' criterion includes both RSS and model complexity and is optimised to find the best α:

$$\mathrm{GCV}(\alpha) = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_\alpha(\mathbf{b}_i))^2}{(1 - M(\alpha)/N)^2}. \tag{4.8}$$

$M(\alpha)$ can be seen as the effective number of parameters used in the model and it accounts for the number of terms as well as the number of parameters used in selecting the optimal positions of the knots. Following J. Friedman et al. (2001) $M(\alpha) = r + c\kappa$, where r equals the number of linearly independent basis functions, κ equals the number of knots selected and c is typically set to 3. It should be noted that it is also possible to use ordinary cross-validation. However, this is very time consuming and preferably uses a lot of training data. The GCV criterion approximates and simplifies this procedure.

Similar to the MLR model the MARS model is trained for each cluster k. As MARS makes use of least-squares, it is also possible to incorporate the fuzzy nature of clustering by means of response weights. As with MLR, all data can be taken into account when performing the MARS procedure and use the observations' membership degrees from $\mathbf{U}_t^k$ as weights in the least-squares minimisation of the MARS routine for cluster k. This is done by premultiplying the observations and sales by the weight matrix $(\mathbf{U}_l^k)^{1/2}$ as explained before.

Readily available *R*-packages can be used in the computational implementation, such as the *mda*-package (Hastie, Tibshirani, Leisch, Hornik, & Ripley, 2016). Results in this research are obtained with the *earth*-package (Milborrow, 2017), which allows observations weights opposed to the *mda*-package.

4.3 Log-transformed models

Both MLR and MARS seek (semi-)linear relations between sales and its predictor variables and build up an additive model. Studies have shown that sales forecasting can also be done using multiplicative models (Kim, Blattberg, & Rossi, 1995). The sales-price dependency – price elasticity – is a well-known example. A multiplicative model can be achieved by transforming sales and its predictive variables by taking logs, and using least squares afterwards. In search of the best forecast, it is therefore investigated whether log-transforms of both MLR and MARS models increase forecasting performance.

Performing a log-linear regression naturally values errors differently than a normal linear regression. With a log-linear regression the ratio between the predicted sales and actual sales is used as error, since $\log \hat{y}_{i,t} - \log y_{i,t} = \log(\hat{y}_{i,t}/y_{i,t})$. This implies that high absolute differences in sales are penalised relatively less compared to lower absolute differences.

Transforming sales has some practical implications. First, it means clustering needs to be carried out with log transformed data as well. This is because the fuzzy clustering distance measure is used in the forecasting procedure, as will be explained in section 4.4. Second, the data contains observations with zero sales, which cannot be transformed directly. One possibility is to set zero sales to a small non-zero value. However, the choice on this non-zero value will influence the forecasting heavily. Therefore, logs are taken after adding 1 to each observation in $\mathbf{X}_t$.

For MLR, this transformation results in a similar regression as (4.1):

$$\log(\mathbf{y}_t + 1) = \tilde{\mathbf{y}}_t = \tilde{\mathbf{B}}_t \tilde{\mathbf{w}}_t + \tilde{\epsilon}_t, \tag{4.9}$$
$$\tilde{\mathbf{B}}_t = \begin{pmatrix} 1 & \log(y_{1,t-1}+1) & \dots & \log(y_{1,t-p}+1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \log(y_{N,t-1}+1) & \dots & \log(y_{N,t-p}+1) \end{pmatrix}.$$

This regression can be used to forecast a product's transformed sales $\log(\hat{y}_{i,t}+1)$, by calculating its expected value $\mathrm{E}[\log(\hat{y}_{i,t}+1)] = \mathrm{E}[\tilde{y}_{i,t}] = \tilde{\mathbf{b}}'_{i,t}\tilde{\mathbf{w}}_t$. However, actual sales are of interest for practical use. This back-transformation involves the following:

$$\begin{aligned} \mathrm{E}[\hat{y}_{i,t}] &= \mathrm{E}[\exp(\tilde{y}_{i,t}) - 1] \\ &= \mathrm{E}[\exp(\tilde{\mathbf{b}}'_{i,t}\tilde{\mathbf{w}}_t + \tilde{\epsilon}_t)] - 1 \\ &= \exp(\tilde{\mathbf{b}}'_{i,t}\tilde{\mathbf{w}}_t)\mathrm{E}[\exp(\tilde{\epsilon}_t)] - 1 \end{aligned}$$

It is important to realise the expectation of the exponent of $\tilde{\epsilon}_t$ is not necessarily equal to the exponent of the expectation as a result of Jensen's inequality. The value of the actual expected sales is approximated by the 'smearing' estimate (Duan, 1983). The extra term in this estimate is the average of the exponential residuals from the regression in (4.9) and proxies the inequality. This method is the same for MARS.

$$\mathrm{E}[\hat{y}_{i,t}] = \exp(\tilde{\mathbf{b}}'_{i,t}\tilde{\mathbf{w}}_t)\left(\frac{1}{N}\sum_{j=1}^{N}\exp \epsilon_{j,t}\right) - 1. \tag{4.10}$$

4.4 Forecasting New Product Sales

Combining the fuzzy clustering algorithm and model specifications leads to the following forecasting procedure for new product sales $\hat{y}_{i,t}$. To verify the methods of Basallo-Triana et al. (2017), the results of this research are mainly based on their procedure:

1. For the set of historical sales $\mathbf{X}_t$, fuzzy clustering with Algorithm 1 yields K clusters;
2. For each cluster k a forecasting model is estimated using sales $\mathbf{y}_t$, observations matrix $\mathbf{B}_t$ and membership grades $\mathbf{U}_t^k$;
3. At time t only observation information is available for a new product i, contained in $\mathbf{b}_{i,t}$. With this observation vector new sales is predicted using all cluster models, resulting in

K forecasts $\hat{y}^k_{i,t}$. This forecast is added to its lagged sales and yields the vectors $\hat{\mathbf{x}}^k_{i,t} = (y_{i,t-p}, \ldots, y_{i,t-1}, \hat{y}^k_{i,t})$;

4. For each $\hat{\mathbf{x}}^k_{i,t}$ the distance measure to its cluster $d^2_{GK}(\hat{\mathbf{x}}^k_{i,t}, \mathbf{c}_k)$ is computed using (3.2);
5. The forecast $\hat{y}_{i,t}$ that yields the smallest value for the distance measure is used as actual forecast, i.e. we choose to follow cluster $\hat{k} = \arg\min_k d^2_{GK}(\hat{\mathbf{x}}^k_{i,t}, \mathbf{c}_k)$.

Alternative Forecasting Procedure

It might seem counter intuitive that this procedure only incorporates one forecasting model in the last step, while fuzzy clustering should allow for model uncertainty using membership grades. In order make fully use of the fuzzy clustering approach, an alteration of this procedure is proposed that replaces step 5 in the forecasting procedure:

5.* Substituting the values of $d^2_{GK}(\hat{\mathbf{x}}^k_{i,t}, \mathbf{c}_k)$ into (3.4a) result in fuzzy weights u_{ik}.

6.* The forecast of a new product is then calculated as the fuzzy weighted average of all individual forecasts $\hat{y}^k_{i,t}$:

$$\hat{y}_{i,t} = \sum_{k=1}^{K} u_{ik} \hat{y}^k_{i,t} \tag{4.11}$$

Note that this approach slightly abuses the original (3.4a), since this is normally used for calculating membership grades of a single observation to each cluster. However, since the value of $y_{i,t}$ is not known in the forecasting effort, it is simply not possible to compute the actual membership grade of a single observation to each cluster. For this same reason, Basallo-Triana et al. (2017) choose a cluster instead of performing a weighted forecast. It can be argued that their procedure does not exploit the full benefits of fuzzy clustering and other clustering approaches could be used as well.

Still, the calculated distance measures carry information on the uncertainty of the individual forecasts. A forecast $\hat{y}^k_{i,t}$ with a relative low distance $d^2_{GK}(\hat{\mathbf{x}}^k_{i,t}, \mathbf{c}_k)$ to the corresponding cluster can be considered relatively certain compared to a forecast with a relatively high distance. Using (3.4a) allows for incorporating this uncertainty by calculating the weights in a fuzzy way. It is easily verified the weights sum to a value of 1.

Chapter 5 Results

Since the methods of Basallo-Triana et al. (2017) are flawed due to the incorrect FK distance measure and erroneous implementation, their forecasting framework is verified while implementing the GK distance measure in the clustering algorithm. In this section the results on forecasting performance are provided and compared to a simple benchmark. Furthermore, it is investigated whether forecasting improves when using MARS instead of MLR or when performing a log-transformation.

First, the evaluation framework and definition of the performance measure is provided. This section then provides results on the following aspects:

1. Tuning clustering and model parameters;
2. Performance of MLR and MARS;
3. Adding more descriptive variables to MARS;

5.1 Evaluation Procedure

The entire forecasting procedure as described in Section 4.4 can be carried out at each time instance $t = 1, \dots, T$, excluding some extra lagged sales used as first variables. In order to provide out-of-sample measurements, the entire data set $\mathbf{X}_t$ is divided into a historical training set and an out-of-sample test set. These sets are denoted by $\mathbf{X}_t^{(H)}$ with $N^{(H)}$ observations and $\mathbf{X}_t^{(T)}$ with $N^{(T)}$ respectively. The sets are divided randomly with a ratio $N^{(H)}/N^{(T)}$ of 80/20. This division is carried out at each time instance t, to flatten out the effect of outliers and make results more stable.

At each time instance, clustering is performed on $\mathbf{X}_t^{(H)}$ and build predictive models for each cluster using the resulting clustering weights. The set $\mathbf{X}_t^{(T)}$ is then used to retrieve $N^{(T)}$ forecasts, which are compared to the actual sales. In total, $N^{(T)} \times T$ forecasts can be compared to their actual sales. These differences are aggregated in the Root Mean Squared Error (RMSE), which is used as performace measure:

$$RMSE = \sqrt{\frac{1}{N^{(T)}} \frac{1}{T} \sum_{i=1}^{N^{(T)}} \sum_{t=1}^{T} (\hat{y}_{i,t} - y_{i,t})^2} \tag{5.1}$$

This measure is absolute, in the sense that it has the same dimensions as the actual sales. To provide a more intuitive measure, the dimensionless relative RMSE is used as well. This computes the error in proportion to the sales:

$$RMSE_r = \sqrt{\frac{1}{N^{(T)}}\frac{1}{T}\sum_{i=1}^{N^{(T)}}\sum_{t=1}^{T}\left(\frac{\hat{y}_{i,t}-y_{i,t}}{\bar{y}_t}\right)^2}, \tag{5.2}$$

$$\bar{y}_t = \frac{1}{N^{(T)}}\sum_{i=1}^{N^{(T)}} y_{i,t}$$

5.1.1 Benchmarking RMSE

All results in this chapter, including tuning parameters and comparing models, are based on forecasting performance in the form of the (relative) RMSE. In order to put the RMSE of various settings and models into perspective, the results are bench-marked to a naive forecast, which uses last period's sales as forecast for current sales. This benchmark does therefore not rely on clustering:

$$\hat{y}_{i,t} = y_{i,t-1} \tag{5.3}$$

Besides putting actual forecasting performance into perspective, the naive benchmark also plays an important role in comparing various models and settings. The setting or model that decreases RMSE most compared to the naive forecast can then be seen as best performing. This way, it is still possible to compare various settings and models.

5.1.2 Notes on RMSE

Some notes on the error measure are in place. First of all, the forecasts are continuous while one can only order a discrete amount of products. The error measure can be adjusted to take this into account by simply rounding the sales forecast to its nearest integer. However, due to the large data set rounding is of little influence on the RMSE.

Secondly, erroneous forecasts may average out in practice due to changes in product stock level. For example, if the forecast of a product is too high at time t and extra stock is added, but is too low at time $t+1$, the forecasting errors had almost no impact. Using this logic, the current error measure – whether or not with rounded forecasts – would probably be biased upwards. The solution to this problem lies in the forecasting procedure, which would need to integrate stock and sourcing information.

Third, it can be justified to favour negative forecast errors opposed to positive errors. This is related to the costs associated with these errors; negative errors can possibly lead to missed profit, while positive errors can lead to obsolete stock due to bad sourcing. If this is the case, the error should be adjusted into an asymmetrical error function. However, this also means the model of choice should be able to handle this as well. This is not within the scope of this research.

Finally, the quantity of products sold is used as variable of interest and measure performance based on this quantity. However, an erroneous forecast has in practice more impact for high-value items. Therefore, it would make sense to adjust the error to nominal sales (€) and alter forecasting models similarly. This is also not within the scope of this research.

5.2 Tuning Clustering and Model Parameters

In order to provide definite results on forecasting performance of all models and settings, it is necessary to tune clustering and model parameters. This includes the amount of clusters K and the number of lags in the MLR p. It is also observed that the RMSE is greatly affected by the products within the data set, especially with respect to the total sales of products in the sample period. All these aspects are optimised using the (relative) RMSE and increase in forecasting performance compared to the naive benchmark. To limit computation time, not every combination of these aspects are considered. Choices on parameters are therefore made on individual behaviour, while keeping other settings constant.

5.2.1 Impact of taking into account highly sold products

In Chapter 2 back-up is provided on the choice of products used in this analysis. However, it turns out that the total sales of products in the entire sample period impacts the forecasting accuracy substantially. This behaviour can be observed in Table 5.1. For an increasing upper bound of total sales, the (relative) RMSE is given for both naive and MLR forecasting, using $K = 6$ clusters and $p = 4$ lags. It can be observed that for both naive and MLR forecasting, a higher upper bound for total sales leads to a higher forecasting error.

A higher absolute RMSE makes sense since products with higher absolute sales are taken into account. However, the relative RMSE tends to increase as well when products with very high total sales are taken into account. Although MLR seems somewhat more robust to this characteristic, the influence of taking highly sold products into account is still present. This behaviour is likely due to the limited number of clusters that can be taken into account. This implies that products with a wide range of sales are combined in one cluster, making it still cucmbersome to fit one model for all observations in a cluster.

For practical use this characteristic implies that popular products cannot be taken into account. Of course, this is problematic since forecasting errors on these products is especially relevant and impactful. However, in practice these products are monitored on regular basis, so expert view can usually cope with these forecasts. This result is used to set an upper bound for the total sales products within the examined data set can have. As it is desired to apply the forecasting methods on the majority of the total data set, the total sales is limited at 1000 over the entire sample period. This reduces the number of products within the data set to 8530, which is used for all further results.

TABLE 5.1: Absolute (relative) RMSE for a range of upper bounds on total product sales. RMSE is calculated for naive and MLR forecasting, using $K = 6$ clusters and $p = 4$ lags.

Total Sales	**Naive**	**MLR**
<250	2.2 (1.8)	1.9 (1.5)
<500	3.2 (1.7)	2.9 (1.5)
<1000	5.4 (1.8)	4.7 (1.6)
<2000	7.8 (2.1)	6.9 (1.8)
<∞	30.7 (3.7)	22.5 (2.9)

5.2.2 Optimal Number of Clusters and Clustering Performance

The number of clusters K needs to be specified as input for the clustering algorithm. The behaviour of increasing K on one-step-ahead forecasting performance is examined, in order to determine the optimal number of clusters. The (relative) RMSE is calculated for MLR forecasting using $K = 2 \ldots 8$ clusters. The MLR model includes $p = 5$ lags and the naive RMSE is calculated again for benchmarking. The results of this analysis are shown shown in Table 5.2.

TABLE 5.2: Absolute (relative) RMSE for MLR forecast using $K = 2 \ldots 8$ clusters. MLR includes $p = 5$ lags and performance is compared to the naive RMSE.

# of clusters	Naive	MLR
2	5.2 (1.8)	4.7 (1.6)
3	5.0 (1.7)	4.4 (1.5)
4	4.9 (1.7)	4.4 (1.5)
5	5.6 (1.9)	5.1 (1.7)
6	5.4 (1.8)	4.9 (1.7)
7	4.9 (1.9)	4.3 (1.6)
8	5.0 (1.8)	4.6 (1.6)

From Table 5.2, it stands out that the naive RMSE fluctuates over different settings of K. This might be counter intuitive, since the naive forecast of (5.3) clearly does not depend in any way on the cluster settings. However, as every RMSE is based on randomly divided test sets, this fluctuation is possible. The naive RMSE is therefore still shown, since it possible to compare various settings by observing the decrease in RMSE of MLR compared to naive forecasting. This is possible because the naive and MLR RMSE of every setting is based on the same test sets. Due to the fluctuation, only the first decimal of RMSE values is provided.

From Table 5.2, it seems the choice of K does not have a strong impact on the forecasting performance of MLR since relative RMSE's are very comparable. This behaviour can have multiple reasons. First, clustering can show bad performance. Even though the number of clusters is increased, it is still possible that most observations are clustered towards only a few clusters. Increasing the number of clusters then proves not useful. An example of this behaviour can be observed in Appendix C. Secondly, over-fitting can become a problem when increasing the amount of clusters. Models are trained on less observations, causing models to be less generalised.

Although results are quite comparable, using $K = 7$ clusters provides best performance. This setting yields the greatest decrease in RMSE of MLR compared to naive forecasting.

5.2.3 Optimal Number of Lags

The number of lags p also influences forecasting performance of MLR. Table 5.3 shows forecasting performance of MLR for a range of lags $p = 1 \ldots 5$. The number of clusters is set to $K = 7$ in this analysis, based on results from the previous section. Again, it is observed that RMSE does not vary substantially when altering the number of lags. However, including $p = 5$ lags is most accurate and provides the highest accuracy gain compared to naive forecasting. In the next section, $p = 5$ lags are used to compare different models.

TABLE 5.3: Absolute (relative) RMSE of MLR forecasting, including a range of lagged sales p. Results are based on $K = 7$ clusters.

# lags	Naive	MLR
1	5.5 (1.9)	5.2 (1.8)
2	5.5 (1.8)	5.6 (1.8)
3	5.9 (1.9)	5.5 (1.7)
4	5.9 (1.8)	5.3 (1.7)
5	5.6 (1.9)	4.8 (1.6)

5.2.4 MLR performance

From the RMSE's used to tune parameters, some conclusions on forecasting with clustering and MLR can be made. Each setting in Tables 5.1, 5.2 and 5.3 leads to a decrease in RMSE compared to naive forecasting. Hence, combining fuzzy clustering with MLR consistently beats naive forecasting.

However, it must be noted that the clustering and MLR approach seems somewhat invariable to parameter settings. Relative RMSE's across all settings are within a small range of values. In the next section a comparison is provided between MLR with tuned settings, i.e. $p = 5$ lags and $K = 7$ clusters, and MARS. It is expected that MARS improves forecasting as it may detect non-linear dependencies in the data.

5.3 Improving Forecasts: MARS & Log Transformations

To improve forecasting, several adjustments are proposed. MARS is investigated as a alternative predictive model, as well as taking log-transformations of the data to investigate multiplicative forms of the models.

5.3.1 Validating MARS and Log Transformations

Naturally, the MARS model can have a different optimum in terms of parameter tuning, e.g. variables taken into account and number of clusters K. Hence, to examine the true difference in forecasting performance, one should first tune the necessary parameters with a validation set – just as carried out in section 5.2 – for all models individually. Then, on a separate test set that is the same for all models concerned, one can calculate the (relative) RMSE and compare results. Naturally, the same thing is true when considering log-transformations of the models considered, described in Section 4.3.

However, as proof of principle and as prospect on the possible performance of MARS, (relative) RMSE of a MARS model is validated. This is done by comparing a MARS and MLR with the same number of lags $p = 5$ and number of clusters $K = 7$. In theory, MARS should be able to result in models comparable to MLR, but with possible non-linear dependencies between current and lagged sales. This should result in comparable (relative) RMSE on the validation set.

This is also carried out for a log-transformation of the data. Note again, this only gives a prospect on actual performance and one should not interpret the results as actual comparison between different models. Results of log transformations and a basic MARS model for $p = 5$ lags and $K = 7$ clusters are shown in Table 5.4.

TABLE 5.4: Absolute (relative) RMSE of naive, MLR and MARS forecasting. Absolute and log-transformed MLR and MARS models include lagged sales as descriptive variables, with up to $p = 5$ lag. $K = 7$ clusters are used in the forecasting process.

Model Type	**Naive**	**MLR**	**MARS**
Absolute	5.2 (1.9)	4.7 (1.6)	4.7 (1.7)
Log-transform	5.6 (1.8)	5.3 (1.7)	5.2 (1.6)

As expected, results of MARS and MLR are comparable when considering the same set of tuning parameters, although MARS performance is slightly worse. This can have several reasons. First, non-linear effects might not be present within the data, so the MARS procedure does not identify any more dependencies as MLR does. Second, the MARS model can be prone to overfitting, so any found dependencies do not generalise to the test set. The GCV criterion of (4.8) should account for this, but the default settings of the *earth*-package are not optimised in this research.

While (relative) RMSE for MLR and MARS are very similar in this setting, computation times are definitely not. For the models presented in Table 5.4, computation time is measured that was necessary to build the models on the training set. This was done for every iteration over time $t = 1 \ldots T$.
On average, MARS needed $67(\pm 37)$ times as much computation time to train a model compared to MLR. On average, MARS needed $22(\pm 20)$ seconds to train a model for one cluster, while for MLR this was only $0.4(\pm 0.3)$ seconds. The large standard deviation in timing differences is mainly due to the varying size of training sets across time iterations.

Unfortunately, timing issues will be even greater when more descriptive variables are added to the model or models are trained with more data. The fact that all available data points are used and observation weights are incorporated does not help in this matter either. The set of splines in (4.6) becomes much larger when using larger data sets and inclusion of more variables, which results in many more function combinations the MARS procedure needs to examine. In practice, these computational burdens would be possibly insurmountable, especially when no severe forecasting improvements are expected.

A first indication for the log transformed models on similar settings ($p = 5$ lags and $K = 7$ clusters) yields that performance is again very similar in terms of relative RMSE, compared to absolute models. Again, this is only a validation and not a full comparison on actual performance differences between MLR, MARS and log transformations. However, it is an indication that log transformations are possible, which is very relevant for using more descriptive variables such as price position, which dependencies with sales are expected to be multiplicative.

5.3.2 Adding more descriptive variables

Because of the timing issues mentioned, it is unfeasible to perform tuning of all parameters and perform a thorough comparison of MLR and MARS. However, a first indication of performance of MARS with more descriptive variables at its disposal is given in this section. This is done for a log-transformed model, since dependencies between sales and e.g. price are expected to be multiplicative, as described in Section 4.3. Moreover, Table 5.4 gives an indication that transforming lagged sales shows no large impact.

Variables that are added to the model are week number, price, price position, delivery time, and percentage of time t a product was actually online. To limit computation time, the number of lags included in the model is 2, although clustering is still carried out on sales profiles including $p = 5$ lags. Limiting the variable space is necessary to decrease computation time. Otherwise, the variable space that MARS needs to cover becomes too large. Results are based on $K = 4$ clusters to reduce computation time even further.

Relative RMSE's for naive, MLR and MARS performance are found to be 1.8, 1.6 and 1.5 respectively. Performance of MLR and MARS is not significantly increased compared to other methods, both in terms of relative RMSE and improvement compared to naive forecasting are similar to any other model and setting.

5.3.3 MLR and MARS Example

All results are based on RMSE's, which are aggregated over all test observations and time instances. However, to provide some extra intuition on model behaviour a specific example an MLR and MARS model is described. For this purpose, a log-transformed MLR and MARS model are trained for a specific time instance t, resulting in $K = 7$ models for both MLR and MARS. In this section some key observations are described, but extended figures and plots are found in Appendix D.

The examples show very bad clustering performance, since models across different clusters can be extremely similar. This is true both for MLR and MARS. Besides, MARS does not show much non-linear dependencies between sales and lagged sales. This result might explain several observations from this section. First, it might be the reason why so little variation in performance is observed while tuning the parameters. If clusters consist of 'garbage', any resulting model will probably be garbage as well. Moreover, this might also explain why the first results of a comparison between MLR and MARS are similar as well, since MARS is simply not able to detect any other significant dependencies in the heterogeneous cluster data.

5.4 Improving Forecasts: Fuzzy Weighted Forecast

All results in this section are based on the forecasting procedure of Basallo-Triana et al. (2017). However, this method chooses a single cluster model for a product. Hence, it does not employ the full potential benefits of fuzzy clustering in which cluster uncertainty is taken into account. This section provides a prospect of accuracy differences between the methods of Basallo-Triana et al. (2017) and the weighted forecast as proposed in Section 4.4.

To this extent, the unweighted (BT) forecast is compared to the fuzzy weighted (FW) forecast. This is done for MLR forecasting with $p = 5$ lags and to exclude any possible dependency on the cluster amount, several values of K are considered. The results are shown in Table 5.5. It is observed that fuzzy weighted forecasts result in very similar (relative) RMSE compared to the methods of Basallo-Triana et al. (2017). The fuzzy weighted forecast consistently show lower RMSE, but this difference is only noticeable in the second decimal.

Unfortunately, the improvements are not as great as expected. It is believed this is due to bad clustering performance. As explained in Section 5.3.3, the models generally show very little difference across the different clusters. This means forecasts with various clustermodels are similar and distances will be similar as well. Incorporating more forecasts therefore is not very effective, only decreasing RMSE slightly. Still, it can be expected that in better data samples, in which clusters are well-separated, a fuzzy weighted forecast improves the methods of Basallo-Triana et al. (2017).

TABLE 5.5: Absolute (relative) RMSE for MLR forecast using $K = 2 \dots 8$ clusters using an unweighted forecast (BT) and a fuzzy weighted forecast (FW), incorporating cluster uncertainty. MLR includes $p = 5$ lags and performance is compared to the naive RMSE.

# of clusters	**Naive**	**MLR-BT**	**MLR-FW**
3	5.0 (1.8)	4.6 (1.6)	4.6 (1.6)
4	4.9 (1.8)	4.4 (1.6)	4.3 (1.6)
5	5.1 (1.8)	4.6 (1.6)	4.5 (1.6)
6	5.5 (1.9)	4.7 (1.6)	4.5 (1.6)
7	5.0 (1.8)	4.6 (1.6)	4.5 (1.6)
8	5.2 (1.8)	4.7 (1.7)	4.6 (1.6)

Chapter 6
Conclusions & Recommendations

This research attempted to verify and extend the methods of Basallo-Triana et al. (2017) to forecast Short Life Cycle Product (SLCP) demand. These methods include fuzzy clustering to cluster analogous sales profiles, after which Multiple Linear Regression (MLR) models are trained on each cluster. A new product is assigned to a cluster and its forecast is calculated with the respective cluster model. Extensions considered are Multivariate Adaptive Regression Splines (MARS) as predictive model, log-transformations of the data and using a weighted forecast that fully exploits the fuzzy nature of the clustering opposed to the methods of Basallo-Triana et al. (2017).

In effort to verify their methods, a theoretical background is provided with a focus on their clustering approach using an improper distance measure. With a different clustering algorithm, forecasting capabilities of analogue based forecasting are investigated using actual sales data of a Dutch e-commerce retailer. This chapter concludes on both of these aspects and provides recommendations for further research on each of these aspects.

6.1 Improper Frigui-Krishnapuram Distance Measure

In their clustering approach, Basallo-Triana et al. (2017) make use of the FK distance measure as defined by Frigui and Krishnapuram (1999). In this research, the fuzzy clustering minimisation is solved for a generalisation of this FK distance measure. With this derivation it is shown that the solution reduces to the simple Euclidean distance, implying that the FK distance measure does not minimise the fuzzy clustering cost function. Moreover, it is observed that the computational implementation of Basallo-Triana et al. (2017) is flawed.

Hence, in this research the original Gustafson-Kessel distance is used, which has a well-known solution to the clustering problem (Gustafson & Kessel, 1979). It must be noted that this algorithm showed some computational challenges, especially with respect to the *fuzzy covariance matrix* and computing its determinant and inverse. This is solved by following Babuska et al. (2002), as solving this issue with the Moore-Penrose inverse still showed ill-behaving convergence.

6.1.1 Recommendations on the Gustafson-Kessel Algorithm

While the GK clustering algorithm has shown to be useful for the identification of hyperplanar clusters in literature, the clustering algorithm requires its user to predefine various parameters. This includes the number of clusters K, the degree of fuzziness q as well as the constraints on the GK distance measure. Only K is actively tuned in this research, while other settings might influence and optimisation over these values can enhance forecasting performance as well. Appendix E shows extra back-up for this conclusion, in which the behaviour of various values of q is examined.

The computational challenges regarding the determinant and inversion of the fuzzy covariance matrix could be solved more properly, as the methods of Babuska et al. (2002) decrease clustering performance. The Moore-Penrose inverse might still show potential, although it must be examined whether the constraints on the GK distance measure can still be maintained when implementing this solution.

Finally, other clustering algorithms might show superior performance compared to the GK algorithm. This includes other (non-fuzzy) clustering approaches, as well as the investigation of other distance measures. The rationale of the improper FK distance measure, e.g. constraining the eigenvalues of the fuzzy covariance matrix, can still make sense, but need a valid mathematical solution to guarantee convergence. In other directions, one might try to find clustering algorithms that optimise the number of clusters in the same algorithm. From a theoretical perspective it might also be interesting to look into possibilities of solving clustering and fitting minimum least squares models in one minimisation. Now, clustering is used merely as a tool to identify clusters of products that share a common prediction model, while minimising actual least squares is the main goal. However, both cost definition depend on the same set of variables.

6.2 Forecasting capabilities MLR and MARS

Like Basallo-Triana et al. (2017), MLR is used as predictive model alongside GK clustering. Performance of this forecasting procedure is investigated with one-step-ahead forecasting of out-of-sample sales and calculation of the relative Root Mean Squared Error (RMSE) of all predictions. These results are compared to a naive benchmark that uses last period's sales as forecast and is independent of the clustering approach. Besides, forecasting improvements are pursued by investigating MARS as predictive model since it is able to detect non-linear dependencies. Also, log-transforms of the models and fuzzy weighted forecasts are investigated as methods to improve forecasting ability.

This research shows for MLR that using $K = 7$ clusters and $p = 5$ lags yields the best performance. For the data set, these settings results in a relative RMSE of 1.6 compared to 1.9 achieved with naive forecasting.

Unfortunately, a full comparison between MLR, MARS and possible log transformations is not carried out due to computational burdens, which causes the tuning of MARS models unfeasible. Computation times for MLR are found to be $67(\pm 37)$ times less compared to MARS. However, a first prospect on forecasting performance show similar results between MLR, MARS and log-transformations. In combination with heavy computational burdens, simple MLR is favoured over any of the other considered models.

Exploiting the fuzzy nature of this clustering approach with a weighted forecast has not shown much potential compared to the methods of Basallo-Triana et al. (2017), in which they assign a product to one cluster model for forecasting. No considerable differences in (relative) RMSE between the two methods are found.

Unfortunately, this decrease in RMSE compared to naive forecasts does not illustrate great potential of analogue based forecasting on the data set used in this research. No large reductions in relative RMSE are observed compared to naive forecasting. Besides, the forecasting procedure seems invariant to tuning, transformation or model choice. Also, models seem to be extremely comparable across clusters. This problem can be caused by the very noisy data set, which causes clustering performance to be very low. It has been observed that clustering still leads to very inhomogeneous groups of sales profiles.

6.2.1 Recommendations on Forecasting Framework

As forecasting performance is likely to be limited by the data set used, further research should focus on data sets that have been proven to work well in forecasting. This way, a solid comparison between the different methods can be given. With a cleaner data set, the clustering performance is possibly improved, so the research can be focused on the proposed extensions.

Of course, this does not mean no improvements can be given on forecasting sales with the data set of this research. Naturally, these improvements rely on handling the data set differently. It has been showed that observations seem to be clustered mainly based on average sales. By normalising each sales profile such that each sales profile attains the same volume might improve forecasting accuracy. Second, forecasting might be carried out for a more contained data set, excluding e.g. observations with almost no sales.

Of course very different forecasting frameworks might proof to be more fruitful, even within the framework of analogue-based forecasting. In this effort, one could use historical sales to assume sales profiles over the entire life cycle of a product. One could then use these parameters as prior guess for a new product in a Bayesian updating framework. This way, information on analogous products can be used, while the impact of other observations is reduces when having enough data of the product itself.

In further practical research on sales forecasting, it might be necessary to include stock levels in forecasts. In practice, up and downward errors might be evened out as stock can balance these errors out. Using this procedure, actual performance in €can be examined. This would be useful in any research regarding sales forecasting.

Appendices

Appendix A
Derivation GK algorithm

In this appendix the solution is derived of the parameters that minimise the fuzzy clustering problem as defined by equations (3.1 - 3.3). This derivation is based on Theodoridis and Koutroumbas (2008) and the original paper of Gustafson and Kessel (1979) on the GK algorithm.

Taking into account the constraints from equations (3.1) and (3.3), a Lagrangian is constructed that needs to be minimised:

$$\begin{aligned}\mathcal{J}(\mathbf{U},\mathbf{A},\mathbf{C}) = \sum_{i=1}^{N}\sum_{k=1}^{K} u_{ik}^{q}(\mathbf{x}_i-\mathbf{c}_k)'\mathbf{A}_k(\mathbf{x}_i-\mathbf{c}_k) \; - \; \sum_{i=1}^{N}\lambda_i\left(\sum_{k=1}^{K}u_{ik}-1\right) \\ -\sum_{k=1}^{K}\beta_k\left(|\mathbf{A}_k|-\rho_k\right) \qquad \text{(A.1)}\end{aligned}$$

In this Lagrangian, λ_i and β_k are the so called Lagrange multiplicators. In this derivation $\mathbf{A}_k$ is assumed to be symmetrical and positive definite. Each parameter u_{ik}, c_k and $\mathbf{A}_k$ is derived by minimising the Lagrangian with respect to the specific parameter and setting it to zero. Furthermore, for brevity $(\mathbf{x}_i-\mathbf{c}_k)'\mathbf{A}_k(\mathbf{x}_i-\mathbf{c}_k) = d^2(\mathbf{x}_i,\mathbf{c}_k)$ is written as d_{ik}^2.

A.1 Membership grades

Minimising equation (A.1) with respect to u_{ik} and setting it to zero yields:

$$\begin{aligned}\frac{\partial\mathcal{J}}{\partial u_{ik}} &= 0 \\ q u_{ik}^{q-1} d_{ik}^2 - \lambda_i &= 0. \\ u_{ik} &= \left(\frac{\lambda_i}{q d_{ik}^2}\right)^{1/q-1}\end{aligned}$$

Using the constraint of (3.1) λ_i is solved, which after a bit of algebra becomes:

$$\lambda_i = \frac{q}{\left(\sum_{l=1}^{K}\left(\frac{1}{d_{il}^2}\right)^{1/q-1}\right)^{q-1}}$$

Substituting this expression for λ_i into the found relation for u_{ik} yields:

$$u_{ik} = \frac{(d_{ik}^2)^{-1/q-1}}{\sum_{l=1}^{K}(d_{il}^2)^{-1/q-1}}$$

A.2 Cluster centroids

For the minimisation of equation (A.1) with respect to $\mathbf{c}_k$, linear algebra states: $\frac{\partial}{\partial \mathbf{s}}(\mathbf{x}-\mathbf{s})'\mathbf{W}(\mathbf{x}-\mathbf{s}) = -2\mathbf{W}(\mathbf{x}-\mathbf{s})$ (Petersen & Pedersen, 2012). In this case, this implies:

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial \mathbf{c}_k} &= \mathbf{0}\\ -2\mathbf{A}_k \sum_{i=1}^{N} u_{ik}^{q}(\mathbf{x}_i-\mathbf{c}_k) &= \mathbf{0}\\ \mathbf{c}_k &= \frac{\sum_{i=1}^{N} u_{ik}^{q}\mathbf{x}_i}{\sum_{i=1}^{N} u_{ik}^{q}}\end{aligned}$$

A.3 Distance transformation matrix

For the minimisation of equation (A.1) with respect to $\mathbf{A}_k$, linear algebra dictates $\frac{\partial}{\partial \mathbf{X}}\mathbf{a}'\mathbf{X}\mathbf{a} = \mathbf{a}\mathbf{a}'$ and $\frac{\partial}{\partial X}|\mathbf{X}| = |\mathbf{X}|(\mathbf{X}^{-1})'$ (Petersen & Pedersen, 2012). Setting the partial derivative to zero results in:

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial \mathbf{A}_k} &= \mathbf{0}\\ \sum_{i=1}^{N} u_{ik}^{q}(\mathbf{x}_i-\mathbf{c}_k)(\mathbf{x}_i-\mathbf{c}_k)' - \beta_k|\mathbf{A}_k|(\mathbf{A}_k^{-1})' &= \mathbf{0}\end{aligned}$$

For brevity $\mathbf{G}_k$ is defined as $\mathbf{G}_k = \sum_{i=1}^{N} u_{ik}^{q}(\mathbf{x}_i-\mathbf{c}_k)(\mathbf{x}_i-\mathbf{c}_k)'$. Also, the symmetric nature of $\mathbf{A}_k$ is used, which implies $(\mathbf{A}^{-1})' = \mathbf{A}^{-1}$. This results in

$$\begin{aligned}\mathbf{G}_k - \beta_k|\mathbf{A}_k|\mathbf{A}_k^{-1} &= \mathbf{0}\\ \mathbf{A}_k &= \beta_k|\mathbf{A}_k|\mathbf{G}_k^{-1}.\end{aligned}$$

To solve for β_k the constraint of $|\mathbf{A}_k| = \rho_k$ is used, as well as the relations $|c\mathbf{X}| = c^n|\mathbf{X}|$ for the $n \times n$ matrix $\mathbf{X}$ and $|\mathbf{X}^{-1}| = 1/|\mathbf{X}|$:

$$\begin{aligned}|\mathbf{A}_k| &= \left|\left(\beta_k|\mathbf{A}_k|\mathbf{G}_k^{-1}\right)\right|\\ \rho_k &= (\beta_k\rho_k)^n\,|\mathbf{G}_k^{-1}|\\ \beta_k^n &= \rho_k^{1-n}|\mathbf{G}_k|\\ \beta_k &= \rho_k^{1/n-1}|\mathbf{G}_k|^{1/n}.\end{aligned}$$

Substituting this expression for β_k into the solution for $\mathbf{A}_k$ yields

$$\mathbf{A}_k = (\rho_k|\mathbf{G}_\mathbf{k}|)^{1/n}\,\mathbf{G}_k^{-1}$$

Most often the fuzzy covariance matrix $\mathbf{F}_k = \mathbf{G}_k/(\sum_{i=1}^{N} u_{ik}^{q})$ is used (see equation (3.4d). With this relation it can be easily verified that the above expression is equal to:

$$\mathbf{A}_k = (\rho_k|\mathbf{F}_\mathbf{k}|)^{1/n}\,\mathbf{F}_k^{-1}.$$

Appendix B
Converge of GK algorithm

As described in Section 3.3, the GK algorithm has problems with inverting the fuzzy covariance matrix $\mathbf{F}_k$ and calculating its determinant. This problem arises when $\mathbf{F}_k$ becomes singular to working precision of the machine, which can also result in a negative determinant.

Two possible solutions for this drawback are proposed, for which converging behaviour is shown of the algorithm with respect to the fuzzy clustering cost function from (3.1). The first solution is proposed by Babuska et al. (2002), which add a scaled identity matrix to $\mathbf{F}_k$ in every iteration of Algorithm 1, while bounding the eigenvalues $\lambda_{k,l}$ of the eigenvalue decomposition $\mathbf{F}_k = \mathbf{E}_k \mathbf{\Lambda}$ as well:

$$\begin{aligned} \mathbf{F}_k^{new} &= \gamma \mathbf{F}_k + (1-\gamma)|\mathbf{F}_0|^{1/\ell}\mathbf{I}_\ell \\ \lambda_{k,l} &= \lambda_{k,\max}/\beta && \forall l \text{ for which } \lambda_{k,\max}/\lambda_{k,l} > \beta \end{aligned}$$

In this research, the values of γ is to 10^{-5} and β is to 10^{15}, so influence on the actual $\mathbf{F}_k$ is small.

The second solution proposes using the Moore-Penrose pseudo-inverse (Petersen & Pedersen, 2012) to invert $\mathbf{F}_k$. We perform this inversion on its eigenvalue decomposition, since this also allows to calculate the determinant of $\mathbf{F}_k$ as well:

$$\begin{aligned} \mathbf{F}_k^- &= \mathbf{E}_k \mathbf{\Lambda}_k^- \mathbf{E}_k \\ |\mathbf{F}_k| &= \prod_{l=1}^{r} \lambda_{k,l}. \end{aligned}$$

The matrix $\mathbf{\Lambda}_k^-$ denotes the pseudo-inverse of the diagonal eigenvalue matrix, which simply implies that all non-zero elements are inverted (zero within working precision). The value r denotes the rank of $\mathbf{\Lambda}_k$. Hence, the determinant of $\mathbf{F}_k$ can be calculated by simply multiplying the non-zero eigenvalues of $\mathbf{F}_k$ (Petersen & Pedersen, 2012).

Figure B.1 shows the cost function value versus hundred iterations of Algorithm 1 for two types of added robustness: following Babuska et al. (2002) and using the Moore-Penrose pseudo-inverse. The same initialisation of values in $\mathbf{U}$ is used for both methods and observations are clustered into 7 clusters. It can be clearly observed that the methods from Babuska et al. (2002) yield a monotonically decreasing cost function value, while implementing the Moore-Penrose pseudo-inverse has convergence issues. It must be noted that the minimum cost function value is actually lower when using the Moore-Penrose method. However, this difference is small enough that reliable converge can be valued more. Therefore, the methods of Babuska et al. (2002) are used instead.

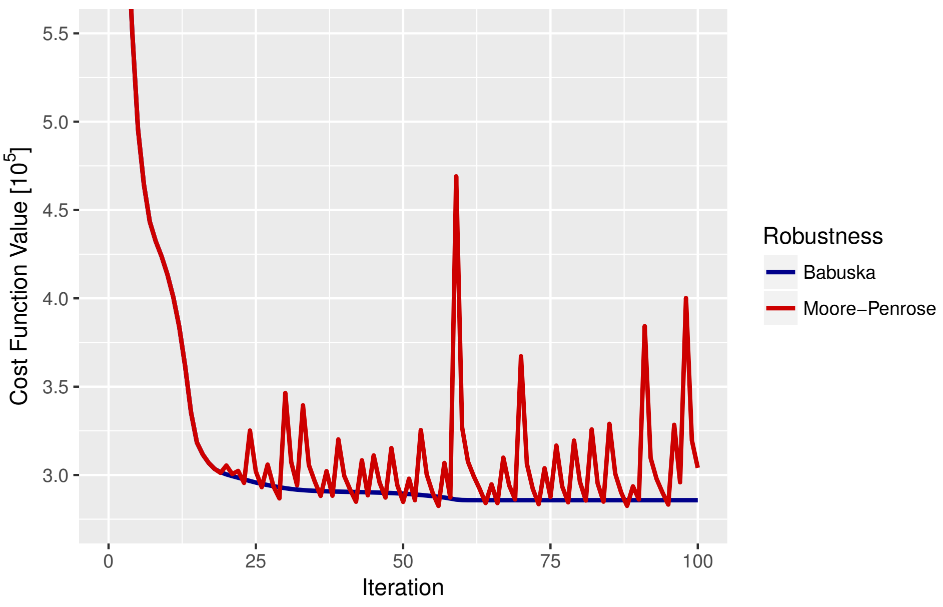


FIGURE B.1: Converge behaviour of GK fuzzy clustering algorithm, implented as Algorith 1. Converge is tested for two robustness measure: Following (Babuska et al., 2002) (blue) and using the Moore-Penrose pseudo-inverse (red).

Appendix C
Clustering Example

This appendix provides a fuzzy clustering example, in which a training set of $N = 6926$ sales profiles on one time instance t are clustered using Algorithm 1. Each observation $\mathbf{x}_{i,t}$ contains 6 weeks of sales, i.e. $\mathbf{x}_{i,t} = (y_{i,t-5}, \dots y_{i,t})'$. The total set $\mathbf{X}_t$ is clustered towards $K = 7$. Other parameters that are used in this clustering are $q = 2$, $\rho_k = 1$ and the algorithm stops iteration over τ when $\max_{i,k} |u_{ik}(\tau) - u_{ik}(\tau - 1)| < 10^{-3}$. These settings are used throughout this entire research. However, an upper bound is not specified regarding the total sales a product in the data set is allowed to have.

The GK algorithm result in grades of membership u_{ik}, denoting the extent to which observation $\mathbf{x}_{i,t}$ belongs to cluster k. To provide some intuition in clustering performance, cluster k is selected for each observation that has the maximum grade of membership. This is the so called 'hard partitioning', resulting in only one cluster for each observation. This result can be seen in Table C.1.

The clustering algorithm does not provide evenly distributed clusters. Especially Clusters 1 and 4 contain very few observations. The cluster centroids are plotted in Figure C.1. As can be seen form this graph, Cluster 1 and 4 contain highest average sales and correspond to the upper two centroids. It seems observations are mainly clustered based on average sales. The actual sales profiles seem to be less distinct. This indicates model building can be meaningless and forecasting improvements compared to naive forecast might not be as high as expected.

For Cluster 7, 34 observations are collected (chosen randomly) and their sales profiles are plotted together with the cluster centroid. The plot can be found in Figure C.2. It seems there is still a lot of variation across sales profiles. This shows the great heterogeneity in the data set, but can also indicate the clustering algorithm finds it hard to distinguish sales profiles across different clusters. This may indicate clustering might be performed with other variables as well, that might be more distinctive features for clustering.

TABLE C.1: Hard partitioning of GK fuzzy clustering algorithm, in which $N = 6926$ are clustered towards $K = 7$ clusters.

cluster k	# of observations
1	4
2	1686
3	1749
4	6
5	1220
6	2116
7	145

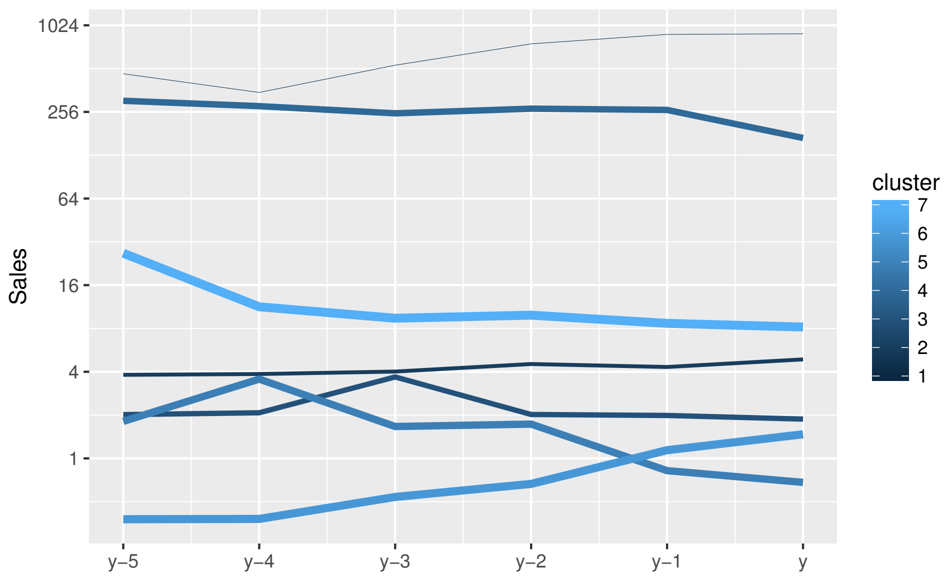


FIGURE C.1: Cluster centroids of GK fuzzy clustering algorithm. Thicker lines mean more observations are partitioned to the respective cluster.

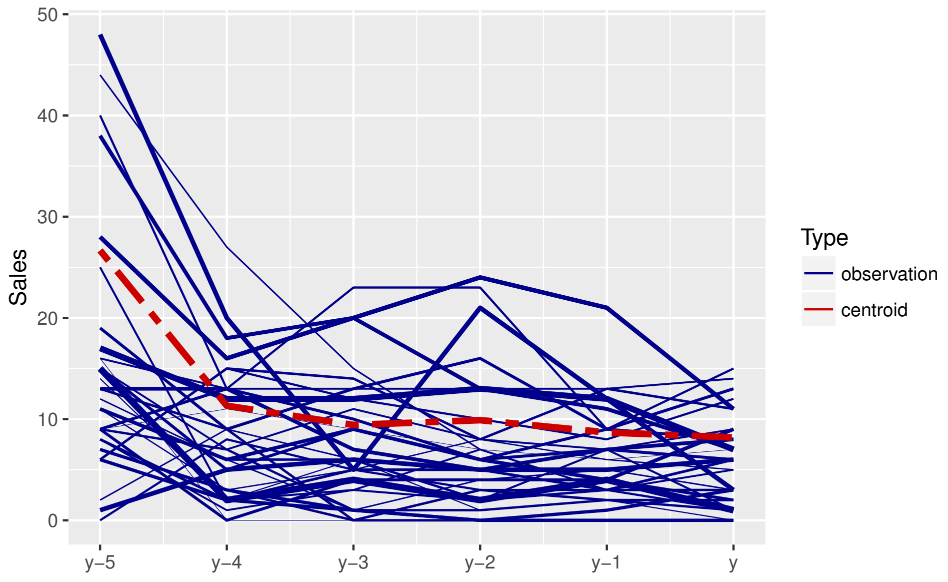


FIGURE C.2: A plot of 34 sales profiles from observations with a hard partitioning to cluster 7. The cluster centroid is plotted in red. The line thickness of each individual observations corresponds to the grade of membership, with a ticker line indicating a higher grade of membership.

Appendix D
MLR and MARS Example

In this section an example is provided on both an MLR and a MARS model. This is used to provide intuition on performance of both methods, specifically with respect to the data set from this research. For both models, extra variables are incorporated such as price and delivery info, similar to the settings in Section 5.3.2.

D.1 MLR

MLR models for $K = 7$ clusters are fitted to the data set. Resulting model parameters are presented in Table D.1. It immediately shows that six out of seven models are almost equal. This is likely due to bad performing clustering, which fails to identify clusters of analogies. Only cluster 4 seems substantially different from other clusters for model training. This difference is mainly caused by an off-set in intercept as it can be seen that other weights are very similar to the other clusters. This is in line with the clustering example of Appendix C, in which it seems that clusters are mainly seperated by average sailes.

TABLE D.1: Fitted model parameters of log-transformed MLR, trained on $K = 7$ clusters with $p = 5$ lags.

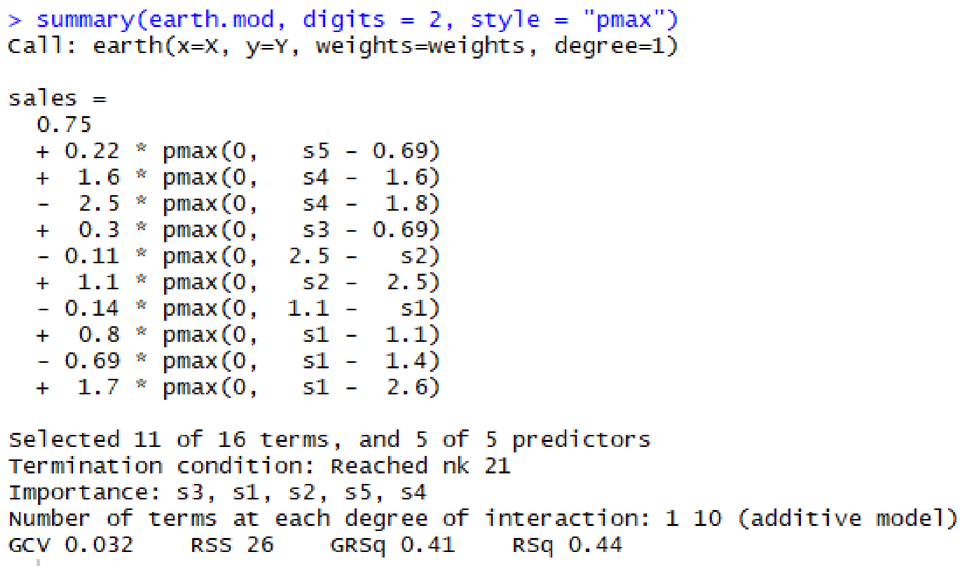
Cluster	w_0	w_1	w_2	w_3	w_4	w_5
1	0.207	0.107	0.021	0.187	0.147	0.253
2	0.206	0.107	0.021	0.188	0.147	0.254
3	0.207	0.107	0.021	0.187	0.147	0.253
4	0.046	0.139	0.052	0.197	0.185	0.272
5	0.207	0.107	0.021	0.187	0.147	0.253
6	0.207	0.107	0.021	0.187	0.147	0.253
7	0.207	0.107	0.021	0.187	0.147	0.253

D.2 MARS

MARS models are trained using the *earth*-package (Milborrow, 2017). This package provides a descriptive summary of the acquired model, as well as several plots to investigate model training, observation residuals and parameter dependencies. These results are presented in Figures D.1 - D.3.

The figures provide some interesting findings. First of all, all lags are used in the MARS model. Second, dependencies show some 'kinks', but in general seem to be quite linear. This can also explain why only minor differences are observed in forecasting performance between MLR and MARS. Third, residuals are not normally distributed and have fat tails. This could be resolved by using generalized linear modeling instead of assuming normal errors.

Finally, different MARS models are very similar across the clusters. This is similar to MLR and might indicate bad clustering performance.



```
> summary(earth.mod, digits = 2, style = "pmax")
Call: earth(x=X, y=Y, weights=weights, degree=1)

sales =
  0.75
  + 0.22 * pmax(0,   s5 - 0.69)
  +  1.6 * pmax(0,   s4 -  1.6)
  -  2.5 * pmax(0,   s4 -  1.8)
  +  0.3 * pmax(0,   s3 - 0.69)
  - 0.11 * pmax(0,  2.5 -   s2)
  +  1.1 * pmax(0,   s2 -  2.5)
  - 0.14 * pmax(0,  1.1 -   s1)
  +  0.8 * pmax(0,   s1 -  1.1)
  - 0.69 * pmax(0,   s1 -  1.4)
  +  1.7 * pmax(0,   s1 -  2.6)

Selected 11 of 16 terms, and 5 of 5 predictors
Termination condition: Reached nk 21
Importance: s3, s1, s2, s5, s4
Number of terms at each degree of interaction: 1 10 (additive model)
GCV 0.032     RSS 26     GRSq 0.41     RSq 0.44
```

FIGURE D.1: Summary of log-transformed MARS model, provided by *earth*-package (Milborrow, 2017). Scales are in logs.

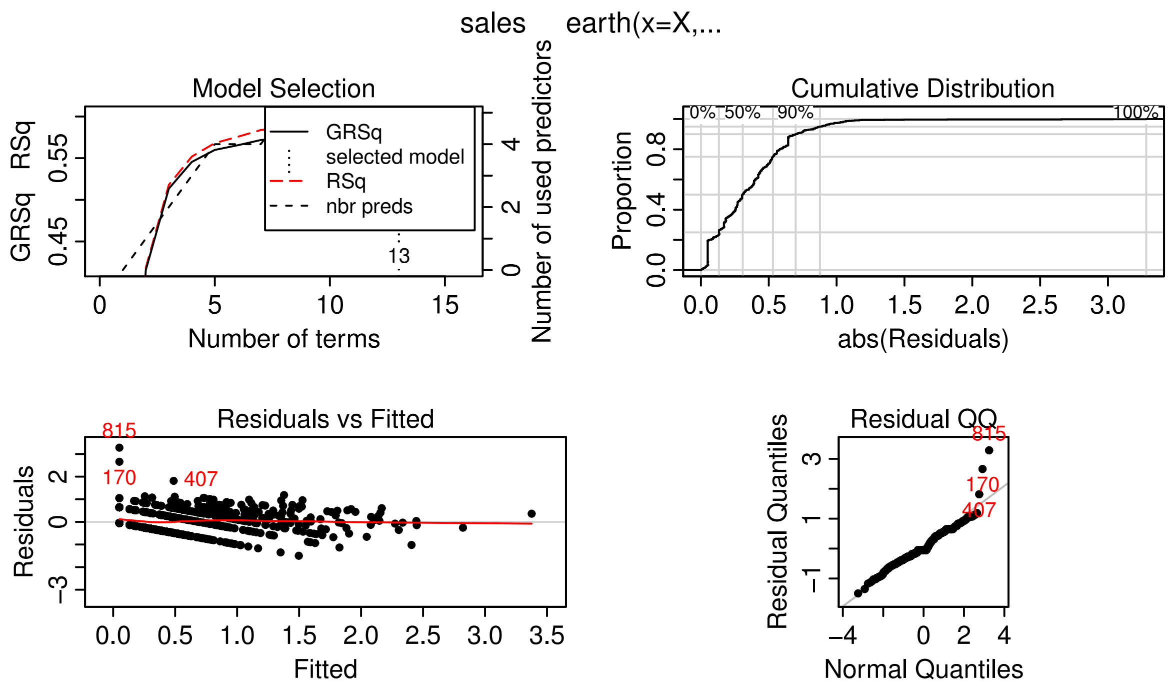


FIGURE D.2: Output of log-transformed MARS model, including GCV plot, RSS plot, QQ-plot of residuals. The output is generated using *earth*-package (Milborrow, 2017). Scales are in logs.

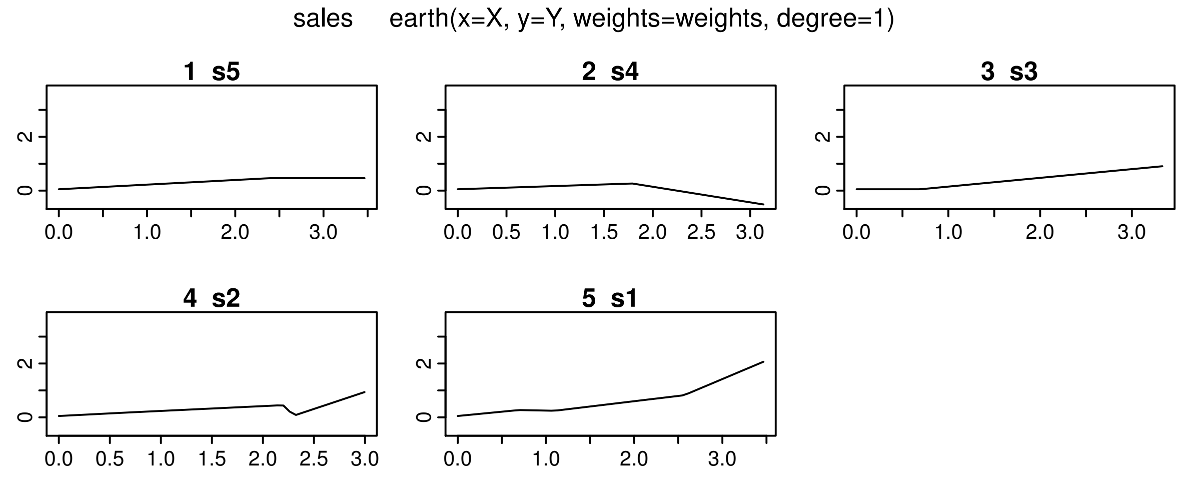


FIGURE D.3: Overview of parameter dependencies, provided by *earth*-package (Milborrow, 2017). Scales are in logs.

Appendix E
Forecasting dependency on fuzziness q

This section provides some extra back up on the forecasting performance for various settings of fuzziness degree q in the GK algorithm. For a range of values for q, the relative RMSE is calculated for MLR forecasting with $K = 7$ clusters and $p = 5$ lags. The results are shown in Table E.1. Naive relative RMSE is calculated as well for benchmarking purposes.

It can be observed that possibly, forecasting performance might be improved for other settings of q opposed to $q = 2$ which is used in this research. Greatest increase in performance compared to the naive forecast is retrieved for $q = 1.7$. Performance increases are very similar across different settings of q, although the limit to crisp clustering yields worse performance. The slight increase in performance compared to $q = 2$ might indicate some marginal increase in forecasting performance can be achieved when q is optimised.

TABLE E.1: Relative RMSE of MLR forecasting with $K = 7$ clusters and $p = 5$ lags. Naive relative RMSE is also shown for benchmarking purposes.

Fuzziness q	Naive	MLR
1.1	1.82	1.85
1.2	1.91	1.73
1.3	1.80	1.68
1.4	1.82	1.71
1.5	1.81	1.69
1.6	1.74	1.59
1.7	1.85	1.59
1.8	1.89	1.67
1.9	1.86	1.66
2	1.76	1.59

Appendix F
R: Gustafson-Kessel Algorithm

```
# This function performs fuzzy clustering using the Gustafson-Kessel algorithm
# The distance measure used is an adapted version of the Mahalanobis distance
#
# Input:
# X: (Nxl) matrix consisting of N observations with l datapoints each
# k: number of clusters in data set
# q: fuzziness parameter (usually set to 2 for fuzzy clustering. 1 = hard clustering)
# tolerance: convergence of the GK algorithm (max dif in consecutive weights U)
#
# Output:
# U: (Nxk) matrix containing k cluster weigths for every (N) observation vector in X
# C: (kxl) matrix containing l cluster centroids for every (k) cluster
# F: (lxlxk) matrix containing k cluster covariance matrices (lxl)
# labels: (Nx1) vector containing 'hard' cluster labels for each observation vector in X
# J: vector (up to max_iter1) of cost function, mimimised using the GK algorithm


GK_cluster_GK <- function(X, k, q, tolerance, max_iter, rho){


  ########################################################################################
  ## Preallocation of all matrices and variables

  N<-nrow(X); #1st dimension of X
  l<-ncol(X); #2nd dimension of X

  U<-array(data=0, dim = c(N,k));        # (Nxk) matrix of fuzzy weights
  F<-diag(l);                            # k (lxl) matrices of cluster covariances
  F<-array(c(rep(F,k)),dim=c(l,l,k));
  A<-F;
  C=array(data=0,dim=c(k,l));            # (kxl) matrix of cluster centroids
  Dist = array(data=0,dim=c(N,k));       # (Nxk) distance for N observations to clusters

  U_prev = array(data=0, dim=dim(U));    # 'previous U', necessary for iteration
  J_old = 1e20;                          # cost function value zero
  J = 0;
  F_cond = array(data=0, dim=c(k,max_iter));
  F_det = F_cond;
  A_det = F_det;

  F_0 = cov(X);
  F_gamm = (det(F_0)^(1/l))*diag(rep(1,l));
  gamm = 1e-5;
  beta = 1e15;
  ########################################################################################






  ########################################################################################
  ## Set up initial guess of U and calculate cluster centroids C, covariances F and D

  t = 1;

  #Generate random sample of U's for which rows add up to 1
  for (i in 1:N){
    rand<-runif(k);
    U[i,]<-rand;
  }
  sum_U<-apply(U,1,sum);
  U<-U/sum_U;
```

```
#First calculation of centroids C, covariances F and distances D based on initial U
for (j in 1:k){
  # Initial Centroids
  U_q = U[, j]^q
  C[j,] = t(U_q) %*% X / sum(U_q)

  #Initial Cluster Covariance
  Xc = X - array(rep(C[j,],each=N),dim=c(N,l))
  U_rep = array(rep(U_q,l),dim=c(N,l))
  F[, , j] <- ( t(Xc) %*% (Xc * U_rep) ) / sum(U_q)

  # Robustness of F through eigen decomposition etc (babuska, 2013)
  F[, , j] <- (1 - gamm) * F[, , j] + gamm * F_gamm

  eig<-eigen(F[, , j]);
  V=eig$vectors;
  D=eig$values;

  sorting=sort(D, index.return = TRUE);
  D=D[sorting$ix];
  D[D[l] / D > beta] <- D[l]/beta;
  V=V[, sorting$ix];

  F[, , j]<- V %*% diag(D) %*% t(V);


  # Calculate A and speed up distance calculation with eigenvalue decomposition
  A[, , j] = ((rho[j] * det(F[,,j]))^(1/l)) * solve(F[, , j]);

  eig<-eigen(A[, , j]);
  E<-eig$vectors;
  Lambda<-eig$values;

  Dist[,j] = ( ( Xc %*% E )^2 ) %*% Lambda;

}

# Calculate cost function value, may be used as extra criteria in while loop.
J[t] = sum( (U^q) * Dist);
#########################################################################################









#########################################################################################
## While loop to minimize J until convergence or max t.
## Inside loop, first the weights in U(t) are updated based on C(t-1), F(t-1), D(t-1)
## Then, C(t), F(t) and D(t) are calculated based on U(t)
## Finally, the value of J (cost function) is evaluated for final check on convergence)

while (max(abs(U-U_prev)) > tolerance & t < max_iter){

  #potential extra while clause:  & J[t]<J_old

  U_prev = U;
  J_old = J[t];


  t = t+1;

  #Update cluster weight matrix U(t)
  dummy_dist<- Dist^(-1/(q-1)); #individual distances to power of -1/q-1
  dummy_sumdist<-apply(dummy_dist,1,sum); # sum all dummy_distances in each row
  U = dummy_dist / array(rep(dummy_sumdist,k),dim=c(N,k));

  #Loop over every cluster to calculate cluster centroids, F and new distances.
```

```
    for (j in 1:k){

      #Calculate new cluster centroids
      U_q = U[, j]^q;
      C[j,] = t(U_q) %*% X / sum(U_q);

      #Calculate new cluster Covariances
      Xc = X - array(rep(C[j,],each=N),dim=c(N,l));
      U_rep = array(rep(U_q,l),dim=c(N,l));
      F[, , j] <- ( t(Xc) %*% (Xc * U_rep) ) / sum(U_q);

      # Robustness of F through eigen decomposition etc (babuska, 2013)
      F[, , j] <- (1 - gamm) * F[, , j] + gamm * F_gamm

      eig<-eigen(F[, , j]);
      V=eig$vectors;
      D=eig$values;

      sorting=sort(D, index.return = TRUE);
      D=D[sorting$ix];
      D[D[l] / D > beta]<- D[l]/beta;
      V=V[, sorting$ix];

      F[, , j]<- V %*% diag(D) %*% t(V);


      # Calculate A and speed up distance calculation with eigenvalue decomposition
      A[, , j] = ((rho[j] * det(F[,,j]))^(1/l)) * solve(F[, , j]);

      eig<-eigen(A[, , j]);
      E<-eig$vectors;
      Lambda<-eig$values;

      Dist[,j] = ( ( Xc %*% E )^2 ) %*% Lambda;

    }

    # Evaluate cost function (function that should be minimized with this algorithm)
    J[t] = sum( (U^q) * Dist);

  } # End while


  #####################################################################################
  ## Calculate 'hard' labels of X for analysis and combine variables to function output.

  labels = apply(U,1, function(x) which(x == max(x)));


  ## Calculate observations per cluster
  cluster_count = array(data = 0, dim = c(k,2));
  cluster_count[, 1] = 1:k;
  for (i in 1:k){
    cluster_count[i,2] = sum(labels==i)
  }


  output <- list(U = U, C = C, F = F, Dist = Dist, A = A, labels = labels,
                 J = J, num_iter = t, cluster_count = cluster_count)



} # End function
```

References

Babuska, R., Van der Veen, P., & Kaymak, U. (2002). Improved covariance estimation for gustafson-kessel clustering. In *Fuzzy systems, 2002. fuzz-ieee'02. proceedings of the 2002 ieee international conference on* (Vol. 2, pp. 1081–1085).

Basallo-Triana, M. J., Rodríguez-Sarasty, J. A., & Benitez-Restrepo, H. D. (2017). Analogue-based demand forecasting of short life-cycle products: a regression approach and a comprehensive assessment. *International Journal of Production Research*, *55*(8), 2336–2350.

Bass, F. M. (1969). A new product growth for model consumer durables. *Management science*, *15*(5), 215–227.

Bass, F. M., Krishnan, T. V., & Jain, D. C. (1994). Why the bass model fits without decision variables. *Marketing science*, *13*(3), 203–223.

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms (advanced applications in pattern recognition)*. Springer.

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, *10*(2-3), 191–203.

Bezdek, J. C., & Hathaway, R. J. (1992). Numerical convergence and interpretation of the fuzzy c-shells clustering algorithm. *IEEE Transactions on Neural Networks*, *3*(5), 787–793.

Davé, R. N. (1990). Use of the adaptive fuzzy clustering algorithm to detect lines in digital images. In *Intelligent robots and computer vision viii: algorithms and techniques* (Vol. 1192, pp. 600–612).

Duan, N. (1983). Smearing estimate: a nonparametric retransformation method. *Journal of the American Statistical Association*, *78*(383), 605–610.

Dunn, J. C. (1973). A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, *3*(3), 32-57.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1). Springer series in statistics New York.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The annals of statistics*, 1–67.

Frigui, H., & Krishnapuram, R. (1999). A robust competitive clustering algorithm with applications in computer vision. *Ieee transactions on pattern analysis and machine intelligence*, *21*(5), 450–465.

Gustafson, D. E., & Kessel, W. C. (1979). Fuzzy clustering with a fuzzy covariance matrix. In *Decision and control including the 17th symposium on adaptive processes, 1978 ieee conference on* (pp. 761–766).

Hastie, T., Tibshirani, R., Leisch, F., Hornik, K., & Ripley, B. (2016). mda: Mixture and flexible discriminant analysis. *R package version 0.4-9, URL http://cran. r-project. org/package= mda*.

Kim, B.-D., Blattberg, R. C., & Rossi, P. E. (1995). Modeling the distribution of price sensitivity and implications for optimal retail pricing. *Journal of Business & Economic Statistics*, *13*(3), 291–303.

Kristof, W. (1969). A theorem on the trace of certain matrix products and some applications. *ETS Research Report Series*, *1969*(1).

Lu, C.-J. (2014). Sales forecasting of computer products based on variable selection scheme and support vector regression. *Neurocomputing*, *128*, 491 - 499. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0925231213008850` doi: http://dx.doi.org/10.1016/j.neucom.2013.08.012

Lu, C.-J., Lee, T.-S., & Lian, C.-M. (2012). Sales forecasting for computer wholesalers: A comparison of multivariate adaptive regression splines and artificial neural networks. *Decision Support Systems*, *54*(1), 584–596.

Milborrow, S. (2017). Package 'earth', multivariate adaptive regression splines. *R package version 4.6.0, URL http://www.milbo.users.sonic.net/earth*.

Petersen, K. B., & Pedersen, M. S. (2012). The matrix cookbook. *http://matrixcookbook.com*.

Radas, S., & Shugan, S. M. (1998). Seasonal marketing and timing new product introductions. *Journal of Marketing Research*, 296–315.

Ten Berge, J. M. (1983). A generalization of kristof's theorem on the trace of certain matrix products. *Psychometrika*, *48*(4), 519–523.

Theodoridis, S., & Koutroumbas, K. (2008). *Pattern recognition, fourth edition* (4th ed.). Academic Press.

Thomassey, S., & Happiette, M. (2007). A neural clustering and classification system for sales forecasting of new apparel items. *Applied Soft Computing*, *7*(4), 1177–1187.

Zhang, G. P., & Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European journal of operational research*, *160*(2), 501–514.

Zhu, K., & Thonemann, U. W. (2004). An adaptive forecasting algorithm and inventory policy for products with short life cycles. *Naval Research Logistics (NRL)*, *51*(5), 633–653.