Multidimensional Prediction Models When the Resolution Context Changes

Adolfo Martínez-Usó(✉) and José Hernández-Orallo

DSIC, Universitat Politècnica de València, Camí de Vera s/n, 46022 València, Spain
{admarus,jorallo}@dsic.upv.es

Abstract. Multidimensional data is systematically analysed at multiple granularities by applying aggregate and disaggregate operators (e.g., by the use of OLAP tools). For instance, in a supermarket we may want to predict sales of tomatoes for next week, but we may also be interested in predicting sales for all vegetables (higher up in the product hierarchy) for next Friday (lower down in the time dimension). While the domain and data are the same, the *operating context* is different. We explore several approaches for multidimensional data when predictions have to be made at different levels (or contexts) of aggregation. One method relies on the same resolution, another approach aggregates predictions bottom-up, a third approach disaggregates predictions top-down and a final technique corrects predictions using the relation between levels. We show how these strategies behave when the resolution context changes, using several machine learning techniques in four application domains.

Keywords: Multidimensional data · Operating context aggregation · Disaggregation · OLAP cubes · Quantification

1 Introduction

Most existing algorithms in machine learning only manipulate data at an individual level (flat data tables), not considering the case of multiple abstract levels for the given data set. However, in many applications, data contains structured information that is multidimensional (or multilevel) in nature, such as retailing, geographic, economic or scientific data. The multidimensional model is a widely extended conceptual model originated in the database literature that can be used to properly capture the multiresolutional character of many data sets [1,5,13,26]. Multidimensional databases arrange data into fact tables and dimensions. A fact table includes instances of facts at the lowest possible level. Each row represents a fact, such as "The sales of product 'Tomato soup 500ml' in store '123' on day '20/06/2014' totalled 25 units". The features (or fields) of a fact table are either measures (indicators such as units, euros, volumes, etc.) or references to dimensions. A dimension is here understood as a particular variable that has predefined (and hopefully meaningful) levels of aggregation, with a hierarchical structure.

A. Appice et al. (Eds.): ECML PKDD 2015, Part II, LNAI 9285, pp. 509–524, 2015.
DOI: 10.1007/978-3-319-23525-7_31

Figure 1 shows several examples of dimensions and hierarchies. Using the hierarchies, the data can be aggregated or disaggregated at different granularities. Each of this set of aggregation choices for all dimensions is known as a *data cube* [6], which provides an easy understanding and offers flexibility for visualisation (aggregated tables and cubes). OLAP technology, for instance, has been developed to handle large volumes of multidimensional data in a highly efficient way, and moving through the space of cubes by the use of roll-up, drill-down, slice&dice and pivoting operators.

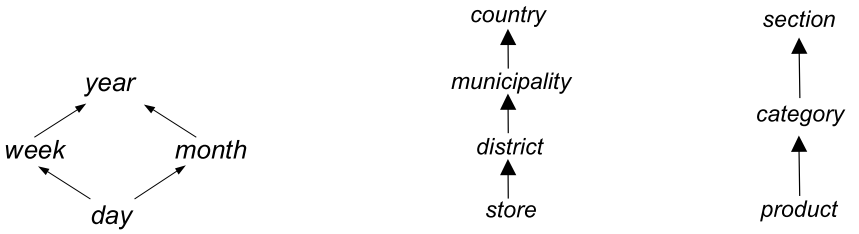

Fig. 1. Examples of dimension hierarchies. Left: Time dimension, Middle: Location dimension, Right: Product dimension.

Despite the success of multidimensional schemas and its widespread use for data warehouses for about two decades, a full integration of machine learning and multidimensional datasets has not taken place. Even in business intelligence tools, which aim at integrating data warehouses, OLAP technology and data mining tools, the usual procedure is to select a cube using an OLAP query or operator, and derive a view from it. Next, this 'minable view' is transferred to the data mining tool to apply machine learning or statistical techniques to this flat, traditional view of the data.

When we analyse the problem more carefully, we see that the main issue for a successful integration is that we would like to use off-the-shelf machine learning techniques but taking full potential of the hierarchical information. Machine learning models are not designed to take hierarchical attributes. Consequently, we need to do something different whenever the cube we want to predict for changes. In other words, the predictions for tomatoes and weeks will be different than the predictions for vegetables and Fridays. These two situations represent *operating contexts*. In principle, a model that has been obtained for one context cannot be *directly* applied to a different context.

This leads us to two major alternatives. On the one hand, we can learn one model for each operating context and apply it for that level of aggregation, which means *retraining* the model for each operating context. On the other hand, we can learn one, more versatile, model at the lowest operating context (highest resolution) and then aggregate their predictions, as in a quantification problem [2,3,11]. This second point of view results in *reframing* the model for each operating context. In addition to these major views, it is worth exploring other models such as *disaggregation*, where a reframing philosophy is addressed

in the opposite way, that is, working at an upper level (lower resolution) and then disaggregating the predictions until the working level of granularity is reached. Finally, there also exists the possibility of *correcting the predictions* worked out for each operating context by means of using the coarse information from upper levels of granularity for improving (correcting somehow) finer predictions, as done by [10] in a multilevel (but not multidimensional) scenario.

In this paper we analyse all these approaches systematically with several machine learning techniques in four different application environments.

The rest of the paper is organised as follows. Section 2 formalises the notion of multidimensional context and properly defines the two main approaches that we will study: the same-level (retraining) approach and the low-level (reframing) approach. Furthermore, the disaggregation model and the same-level correction model are also defined in this section. Section 3 discusses how datamarts have to be understood when models are required to predict some of the measures of the fact table and also states some measurement considerations. Section 4 presents the techniques, datamarts and error measure that will be used in the experiments. After that, results for each approach are analysed. Section 5 discusses some related work and section 6 closes the paper with some take-away messages and some future work.

2 Multidimensional Contexts

We consider a multidimensional data set D (or datamart) of schema $\langle X, Y \rangle$ where $X = \{X_1, \ldots, X_d\}$ is the set of d dimensions (used as predictor attributes or features) and Y, which is the *target attribute* (one measure or indicator that can be numeric or nominal). We use D_A to denote the projection of dataset for attribute A. Note that datasets and projections are multisets (i.e., they can have repeated values). Each dimension X_i has an associated hierarchy $h(X_i)$ of m_i elements or *levels* $\{X_i^{(1)}, \ldots, X_i^{(m_i)}\}$ with a strict partial order $<$. In this paper we will assume that hierarchies are linear, so the partial order becomes a total order from the lowest level $X_i^{(1)}$ to the highest level $X_i^{(m_i)}$. This is not a strong restriction, as a non-linear dimension can be converted into several linear dimensions (one for each possible pathway in the lattice). For instance, if $X_2 = \mathsf{location}$, as in Figure 1 (middle), we have $X_2^{(1)} = \mathsf{store}$, $X_2^{(2)} = \mathsf{district}$, $X_2^{(3)} = \mathsf{municipality}$ and $X_2^{(4)} = \mathsf{country}$ with $\mathsf{store} < \mathsf{district} < \mathsf{municipality} < \mathsf{country}$ and their transitive closure. We will consider that the top level m_i for every hierarchy is $\mathsf{all\text{-}i}$, such that for every $l \in h(X_i)$, $l < \mathsf{all\text{-}i}$. Non-hierarchical attributes are just special cases, by just considering that $m_i = 2$ (the bottom and the top $\mathsf{all\text{-}i}$ level). These dimensions then just become regular attributes but with the possibility of aggregating them to the top level all.

Each level $X_i^{(j)}$ of a hierarchy $h(X_i)$ has an associated domain $\mathcal{X}_i^{(j)}$, which can be nominal or numeric. We will assume that there are no levels with the same name in the same or different hierarchies. In this way, if the name of a level is name then we can just refer to the level by $X^{(\mathsf{name})}$ and the associated domain

by $\mathcal{X}^{(\mathsf{name})}$. For instance, the domain of the level country for dimension location, i.e., $\mathcal{X}_2^{(4)}$, or $\mathcal{X}^{(\mathsf{country})}$, might be the set with values $\{\mathtt{UK}, \mathtt{Spain}, \mathtt{France}\}$. For every pair of consecutive levels $X_i^{(j)}$ and $X_i^{(j+1)}$ in a hierarchy we define a regrouping function ϕ_i^j between the values of $X_i^{(j)}$ to the values of $X_i^{(j+1)}$. For instance, $\phi_2^3(\mathtt{Valencia}) = \mathtt{Spain}$. We denote by $\phi_i^{j:k}$, with $j \leq k$ the successive application of ϕ from j to k, i.e., $\phi_i^{j:k}(v) = \phi_i^k(...\phi_i^{j+1}(\phi_i^j(v))...)$. Given a value v at a level $X_i^{(k)}$ of the dimension i, we denote by $\perp(v)$ the set of all the values at the lowest level of that hierarchy that belongs to, i.e., $\{w \in \mathcal{X}_i^{(1)} \mid \phi_i^{1:k}(w) = v\}$. For instance, $\perp(\mathtt{Valencia})$ would be all the stores of all the districts of Valencia.

Definition 1. *A multidimensional operating context or resolution is a d-tuple of levels $\langle l_1, \ldots, l_d \rangle$, with each $l_i \in h(X_i)$. A multidimensional context determines the level for every dimension of the dataset.*

Definition 2. *Given a multidimensional context, a selection of D at a context $\langle l_1, \ldots, l_d \rangle$ with values $\langle v_1, \ldots, v_d \rangle$ is defined as follows:*

$$\sigma_{[l_1=v_1,\ldots,l_d=v_d]}(D) \triangleq \{\langle x_1, \ldots, x_d, y \rangle \,|x_1 \in \perp(v_1), \ldots, x_d \in \perp(v_d)\} \qquad (1)$$

For instance, if we have three dimensions $X_1 = \mathsf{product}$, $X_2 = \mathsf{location}$ and $X_3 = \mathsf{time}$, and Y is representing units, we could select all the facts for the context $\langle \mathsf{item}, \mathsf{municipality}, \mathsf{year} \rangle$ with values `tomato`, `Valencia` and `2013` respectively with $\sigma_{[\mathsf{item}=\mathtt{Tomato},\mathsf{munic.}=\mathtt{Valencia},\mathsf{year}=\mathtt{2013}]}(D)$.

Finally, we can define an aggregation operator as follows:

Definition 3. *Given an aggregation function, agg, as a function from sets to real numbers, the aggregation of a datamart D for a context $\langle l_1, \ldots, l_d \rangle$ is defined as follows:*

$$\gamma^{agg}_{[l_1,\ldots,l_d]}(D) \triangleq \{\langle x_1, \ldots, x_d, z \rangle \,|x_1 \in \mathcal{X}^{(l_1)}, \ldots, x_d \in \mathcal{X}^{(l_d)}, \\ z = agg(\{y| \, \langle v_1, \ldots, v_d, y \rangle \in \sigma_{[l_1=x_1,\ldots,l_d=x_d]}(D)\})\}$$

The above aggregation is extended for unlabelled datasets with no y attribute.

For instance, $\gamma^{sum}_{[\mathsf{item},\mathsf{municipality},\mathsf{year}]}(D)$ returns all the tuples for each possible combination of values at the level item in the dimension product, at the level municipality in the dimension location and at the level year in the dimension time, where the output variable is constructed by summing all the y of the corresponding rows according to the hierarchies.

Given the above notation, now we consider a predictive problem from X to Y. For instance, how many tomatoes we expect to sell in Valencia next week? Assuming we have a training dataset, how would we train our model? As a first idea, it seems reasonable to aggregate the training data using the γ operator above for the context $c = \langle \mathsf{item}, \mathsf{municipality}, \mathsf{week} \rangle$, producing a model M

that will be applied to the deployment data with the same context. However, if some time later we are interested in predicting sales for all vegetables for next Friday, what would we do? We could aggregate the training data for the context $c' = \langle \mathsf{category}, \mathsf{municipality}, \mathsf{day} \rangle$, learn a new model M' and predict for the deployment data. This is what we see in Figure 2 (top). We refer to this approach as the *Same-Level* (SL) approach or the *retraining* approach.

Definition 4. *Given a training data T with measure Y and a deployment data D, a model learnt for measure Y at the* same level*, denoted by* SL*, in context $\langle l_1, \ldots, l_d \rangle$ is defined as follows. We first aggregate T for that context, i.e., $T^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(T)$. Then we train a model $M^{\Delta} : X^{\Delta} \rightarrow Y^{\Delta}$. For the deployment data D we also aggregate the original data as $D^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(D)$. We finally add an attribute $\hat{Y}$ to D^{Δ} by setting it equal to the predictions of the model M^{Δ} for each of these aggregated rows, so producing $\hat{D}^{\Delta}$. This yields pairs $\langle X, \hat{Y} \rangle$ at that context.*

An alternative approach goes as follows. Consider that we train a predictive model M for the lowest level in D. Once a new multidimensional appears, we apply the model to the deployment data and *aggregate the predictions*. With this approach, one model is used for every possible context. This is illustrated in Figure 2 (middle). We refer to this approach as the *Lowest-Level* (LL) approach or the *reframing* approach.

Definition 5. *Given a training data T with measure Y and a deployment data D, a model learnt for measure Y at the* lowest level*, denoted by* LL*, and deployed at a context $\langle l_1, \ldots, l_d \rangle$ is defined as follows. We first train a model $M : X \rightarrow Y$ for the whole training dataset T. Now, for each row at the lowest level in the deployment data D we apply M. We add a new attribute $\hat{Y}$, and set it to the result of the model for each row, giving a new dataset $\hat{D}$. Finally, given an aggregation function agg, we now calculate the predictions for a context $\langle l_1, \ldots, l_d \rangle$ as $\hat{D}^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(\hat{D})$, which produces pairs $\langle X, \hat{Y} \rangle$ at that context.*

Another alternative is to disaggregate predictions from a higher level of granularity. Figure 2 (bottom), trains a predictive model M for a higher level and keeps the frequencies or proportions that are shared on the lower levels for the training set. Then with a new prediction at a higher level, the frequencies are used for the disaggregation. We refer to this approach as the *Disaggregation* (dAg) approach. Figure 3 shows a disaggregation example taking into account the two first levels of the product dimension from Fig.1. This example takes into account one dimension and only one level of disaggregation. However, it could be extended to more than one level of disaggregation, for instance from section to product in the same dimension (2 levels), and for more than one dimension, for instance taking into account dimensions product and location, which would also imply working out all the combinations. Nonetheless, for simplicity, in the rest of the paper, we have limited the disaggregation approach to the following setting:

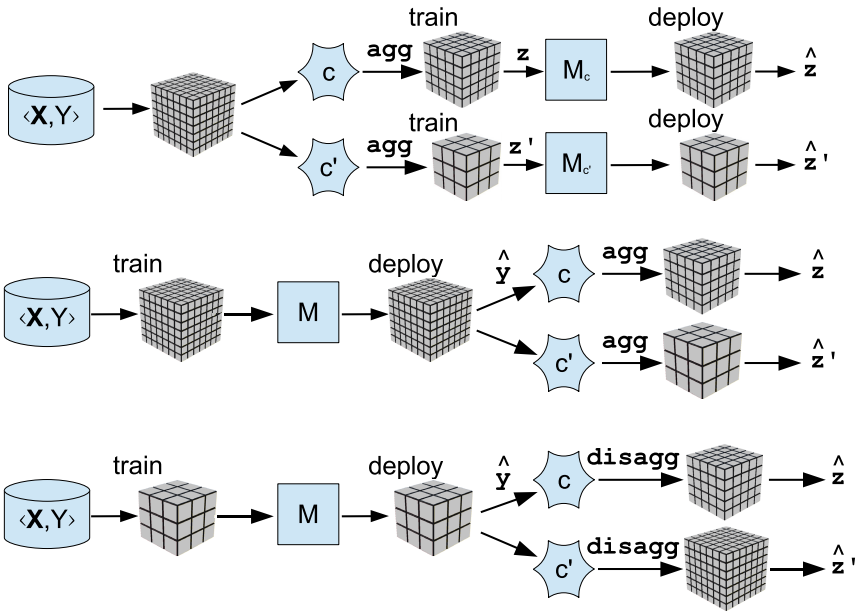

Fig. 2. Retraining (*Same-Level* approach), reframing (*Lowest-Level* approach) and disaggregation for two different multidimensional contexts c and c'. Retraining (top) needs to convert the training data for the two contexts c and c' and then aggregating the output into z or z' respectively. Two models M_c and $M_{c'}$ are learnt (one for each context) at the same level the predictions must be done. Reframing (middle) shows how the training data is used just once at the lowest level to create a single model M that is applied to different operating contexts c or c' by aggregating the outputs appropriately. Disaggregation (bottom) shows how the training data is used just once at a higher level to create a single model M that is applied to different operating contexts c or c' by disaggregating the outputs appropriately.

- Only one level per dimension has been disaggregated for each result, that is, we just disaggregate to the level immediately below.
- All the dimensions are taken into account, although we only disaggregate one dimension for each result.

Definition 6. *Given a training data T with measure Y and a deployment data D_{lo}, a model learnt for measure Y using* disaggregation*, denoted by* dAg*, and deployed at a context $\langle l_1, \ldots, l_d \rangle$, is defined as follows. Let us suppose the context $\langle l'_1, \ldots, l'_d \rangle$ as the context at the level immediately below to $\langle l_1, \ldots, l_d \rangle$. We first aggregate T for the upper context, i.e., $T^{\Delta}_{hi} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(T)$ and for its immediately context below, i.e., $T^{\Delta}_{lo} \triangleq \gamma^{agg}_{[l'_1,\ldots,l'_d]}(T)$. We define a function F that counts the number of observations that fall into each of the disjoint categories. This is done as an aggregation function as in Definition 3 but using count as the aggregation function. Let us also suppose that we have predictions for the deployment data D^{Δ}_{hi} at the high level using the* SL *approach. Now, for each row at the lower*

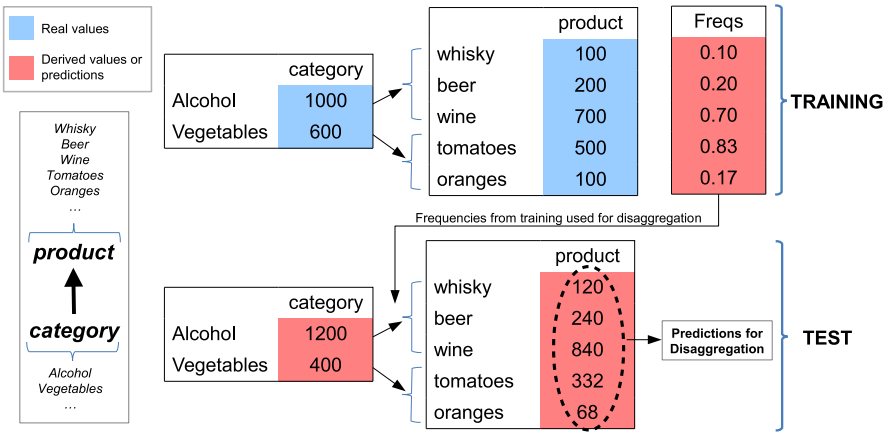

Fig. 3. Disaggregation example for the product dimension. The frequency of each product (whisky, beer, tomatoes, etc.) within each category (Alcohol, Vegetables) is learned during the training. These frequencies are then applied to the predictions made for a higher level of granularity resulting in the disaggregated predictions for the lower level. Light blue cells represent real values whereas light red represent derived values (such as frequencies) or predictions.

context in the deployment data D_{lo}^{Δ} *we apply* F *on the predictions for* D_{hi}^{Δ} *to create the predictions for the lower level.*

Finally, in [10], the authors presented a method for improving the current predictions using the coarse information from upper levels of granularity. Their methodology uses the approximation values of the aggregated targets (in our case the predictions) and the predictions of the individual targets for producing new modified predictions. Figure 4 shows an example of this procedure taking into account the two first levels of the product dimension from Fig.1. As it can be seen, we work out a correction ϵ as the relation between the sum of the predictions at the category level and the sum of the predictions at the product level, using then this value for uniformly distributing the differences among the predictions of the lower level. This approach is actually a correction of the same-level model and we thus refer to this approach as the *same-level correction* model (SLc).

Definition 7. *Let us consider the deployment data at two different contexts,* D_{lo} *and* D_{hi}*, where deployment data* D_{lo} *is defined at the context immediately below to* D_{hi}*.* $\hat{D}_{lo}^{\Delta}$ *and* $\hat{D}_{hi}^{\Delta}$ *have been produced applying the* SL *model to each deployment data respectively and, therefore,* $\hat{Y}_{lo}$ *and* $\hat{Y}_{hi}$ *attributes with predictions can be found for each context respectively. Thus, the* same-level correction *model, denoted by* SLc*, is defined as follows. We first work out* $s\hat{Y}_{lo} = \sum_i \hat{Y}_{lo}[i]$ *and* $s\hat{Y}_{hi} = \sum_i \hat{Y}_{hi}[i]$ *and let us define* $\epsilon = \frac{s\hat{Y}_{hi}}{s\hat{Y}_{lo}}$*. For the deployment data* D_{lo} *we correct the attribute* $\hat{Y}_{lo}$ *for each row by means of multiplying each* $\hat{Y}_{lo}$ *value by* ϵ*.*

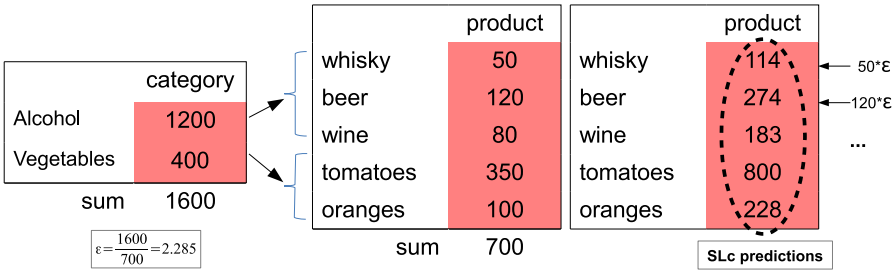

Fig. 4. Same-level correction example for the product dimension. Cells in light red represent predictions.

3 Measure Properties and Mean Models

The first thing we need to consider is the kind of machine learning tasks that are common with multidimensional data. The way the information is arranged in a multidimensional schema, with a fact table containing *measures* suggests that many machine learning tasks, especially predictive ones, are usually focussed on predicting the measures. For instance, if facts are sales, consumptions, failures, usages, etc., it is common to become interested in predicting some of the measures in these tables (e.g., units, dollars, hours, etc.) from past data. As measures are usually numerical, many problems will turn out to be regression problems. Nonetheless, some measures can be nominal, such as whether a purchase has been satisfactory or not. In that case, however, the measure becomes a percentage, i.e., a number, when we aggregate, so binary nominal measures can also be taken as numbers.

The time dimension is found in most datamarts. In a predictive scenario, the time dimension becomes slightly special: predictions are about future facts, so training is usually performed with available data up to a given time and the model is then used to extrapolate from that point on (next week, next month, next year, depending on the resolution). This occurs in three out of four datamarts used in this work. Some other databases are sparse and only collect positive cases (e.g., 0 sales are not included) and have to be preprocessed to contain this information for them to be meaningful.

Another important issue about multidimensional schemas is whether the measures we want to predict are *additive*, *semi-additive* or *non-additive*. A measure is *additive* when, for any dimension and any set of values S at level j that we want to aggregate up to level $n+k$, the summation of these values using any partition of S is equivalent, i.e., gives consistent results. For instance, units sold in a supermarket aggregate well for all dimensions, i.e., the result is independent of the way it has been aggregated. However, percentages do not aggregate well, as the denominator is not known when performing the aggregation. Therefore percentages are *non-additive*. Finally, the term *semi-additive* measure is used for those measures that aggregate well for some dimensions but not for others. For instance, measures that accumulate or depend on the state, such as stock levels are usually semi-additive.

The aggregated function that is used for aggregating datamarts, as in Definition 3, does not have to always be $sum(S) \triangleq \sum_{s \in S} s$. For instance, it could be an average, $avg(S) \triangleq \frac{sum(S)}{|S|}$. Some functions just work for some measures. For instance, consumption (e.g., in kWh) can be aggregated by averaging it. However, we have to be very careful about how this aggregation is performed. For instance, *avg* is not *composable*.

In regression tasks, we usually look at a baseline method that consists in averaging the values for the training data and apply these values systematically during deployment. This is known as the *mean* or *constant* model. In this work, we define our baseline method as follows:

Definition 8. *Given a training data T with measure Y and a deployment data D, the MEAN model for measure Y at the* same level *in context $\langle l_1, \ldots, l_d \rangle$, denoted by SL.MEAN, is defined as follows. We first aggregate T for that context, i.e., $T^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(T)$. Then we calculate $\overline{Y^{\Delta}} \triangleq avg(T^{\Delta}{}_Y)$, the average of the measure Y for this context. For the deployment data D we also aggregate the original data as $D^{\Delta} \triangleq \gamma^{agg}_{[l_1,\ldots,l_d]}(D)$. We finally add an attribute $\hat{Y}$ to D^{Δ} by setting it equal to $\overline{Y^{\Delta}}$ for every row in D^{Δ}, so producing $\hat{D}^{\Delta}$. This produces pairs $\langle X, \hat{Y} \rangle$ at that context.*

4 Experimental Setting and Results

The MEAN approach is useful as a baseline, but we of course are interested in the use of machine learning methods to get good predictions. We have also considered other four techniques: LRW (linear regression using RWeka in R [14,22]), M5P (regression tree using RWeka), SVM (package `e1071` in R, linear kernel) and KNN (package `kknn` in R). The datamarts used are now presented:

- **GENOMICS**: Originally, this human genome dataset contains genomic data (HGDB) from several public and private research databases, including information about genes, chromosomes, mutations, diseases, etc. structured in 20 (numerical and nominal) attributes [20]. Data goes from years 1970 to 2012 and the output variable is the number of variations. We converted it into a multidimensional datamart, where each fact showed the number of variations according to five different dimensions (hierarchies in parenthesis): SPEC (Eff $<$ All), GENOTYPE (ID $<$ Chrom $<$ All), PHENOTYPE (Name $<$ ICD10 $<$ ICD10.Cat $<$ All), DBANK (Dbnk $<$ All) and DATE (Year). Note that as we use the DATE dimension to split the data we only consider one level here. The number of possible multidimensional contexts is then $2 \times 3 \times 4 \times 2 \times 1 = 48$.
- **AROMA**: This is an artificial dataset constructed from IBM sales information. It contains sales data for coffee and tea products sold in stores across the United States [16]. The data is almost directly converted into a multidimensional datamart where each fact describes the sales of products using two measures (units and dollars, although we will only use dollars as the

output variable) according to five dimensions (hierarchies in parenthesis): PROMO (KeyPromo < PromoType < All), CLASS (KeyClass < All), PRODUCT (KeyProduct < All), STORE (KeyStore < KeyMKT < MKT-HQ-City < MKT-HQ-State < MKT-District < MKT-Region < All) and PERIOD (Year). Note that as we use the PERIOD dimension to split the data we only consider one level here. Data goes from years 2004 to 2006 and the number of possible multidimensional contexts is $3 \times 2 \times 2 \times 7 \times 1 = 84$.

- **CARS**: This is a dataset for car fuel consumption and emissions which is created as a reduced representation of [9] (some attributes are removed) in order to construct a datamart. It describes fuel consumption in cars from years 2000 to 2013, being published by the UK's Vehicle Certification Agency (VCA). The target variable is car fuel emissions (CO_2) and we have six dimensions (hierarchies in parenthesis): CAR (Man.Model.Description < Man.Model < Manufacturer < All), ENGINE (EngineCapacity < All), TRANS (Transmission < TransType < All), EURO (EuroSTD < All), FUEL (FuelType < All) and TIME (Year). Note that as we use the TIME dimension to split the data we only consider one level here. The number of possible multidimensional contexts is $4 \times 2 \times 3 \times 2 \times 2 \times 1 = 96$.
- **UJIndoorLoc**: This is a dataset for benchmarking indoor localisation algorithms [25]. UJIndoorLoc contains the Wi-Fi access points readings for all the spaces (offices, laboratories, etc.) of the School of Technology and Experimental Sciences of the University Jaume I. The average signal intensity has been used as the target variable ($INTENSITY$). We have three dimensions in this case (hierarchies in parenthesis):WHERE (Space < Floor < Building < All), TIME (Hour < Day < WeekDay < All), PHONE (Model < Manufacturer < All), being the number of possible multidimensional contexts $4 \times 4 \times 3 = 48$.

We split GENOMICS, AROMA and CARS datasets into training and test on the basis of a *split-year*. Parameter *split-year* has been set to 2006 for all the datasets, being the *split-year* included in the test set. On the other hand, for UJIndoorLoc dataset, the authors already provide the training and evaluation subsets (see [25]). No rows with zeros were added to CARS and UJIndoorLoc datasets, as missing cases are just absence of information. The target variable is a ratio, so the aggregation function that makes sense for these datasets is *avg*, which is neither additive nor associative. Finally, we clip the predictions of all methods to 0 if they are negative, as in the four datamarts the measures cannot be negative. This is important for methods that could potentially predict negative values such as M5P or LRW.

As the four datamarts have led to regression problems, we may use the Mean Squared Error (*MSE*) as the error measure. However, for the two datamarts that use *sum* as aggregating function, the magnitude of the error will be much higher for highly aggregated contexts, and the values will be difficult to compare. A good way of getting rid of this problem is to divide *SE* (or *MSE*) by the *SE* (or *MSE*) of the MEAN model. Interestingly, in a classical regression setting, the *MSE* of the MEAN model equals its error variance. So, actually, what we are

doing is to show the *MSE* by some kind of error variance. We use the SL.MEAN model, as it ensures that it is constant for the deployment multidimensional context.

Definition 9. *The normalised squared error (NSE) of a technique* TECH *is defined as* $\frac{MSE(\mathsf{TECH})}{MSE(\mathsf{SL.MEAN})}$.

Table 1. Comparison *NSE* among LL, SL, dAg and SLc.

	GENOMICS				AROMA				CARS				UJIndoorLoc			
	SL	LL	dAg	SLc	SL	LL	dAg	SLc	SL	LL	dAg	SLc	SL	LL	dAg	SLc
MEAN	1.00	0.57	0.46	0.62	1.00	0.54	0.28	0.53	1.00	0.92	1.39	1.73	1.00	0.97	2.51	6.99
SVM	0.49	0.50	0.44	0.63	0.11	0.03	0.02	0.05	0.73	0.46	1.25	1.64	1.02	1.21	2.49	7.00
M5P	0.92	0.14	0.42	0.85	0.87	0.04	0.34	0.40	0.79	0.77	1.31	1.62	1.06	2.81	2.49	7.07
LRW	1.02	0.58	0.44	0.82	1.09	0.67	0.33	0.56	0.84	0.92	1.31	1.66	1.07	0.82	2.49	7.05
KNN	0.87	0.08	0.36	0.71	1.07	0.03	0.27	0.34	0.79	0.48	1.51	1.88	1.08	1.31	2.53	6.95

Table 2. Rank summary for the four datasets.

	GENOMICS				AROMA				CARS				UJIndoorLoc			
	SL	LL	dAg	SLc	SL	LL	dAg	SLc	SL	LL	dAg	SLc	SL	LL	dAg	SLc
MEAN	5.00	2.80	1.53	3.97	5.00	3.22	1.97	3.57	2.37	1.66	3.86	2.62	1.88	1.49	3.18	3.81
SVM	3.48	3.31	1.58	4.83	4.34	3.37	2.11	3.71	2.48	1.06	3.73	3.33	1.38	2.39	2.81	3.85
M5P	4.74	1.00	2.80	4.02	5.00	1.01	2.95	3.83	2.35	1.83	3.67	2.78	1.34	3.24	2.38	3.65
LRW	4.98	2.66	1.71	3.94	5.00	3.92	1.89	3.01	2.07	2.48	3.50	2.67	2.30	1.09	3.05	3.96
KNN	4.58	1.00	2.53	4.28	5.00	1.00	2.88	3.74	2.20	1.15	3.84	3.40	1.43	2.28	2.93	3.81
Overall	4.56	2.15	2.03	4.21	4.87	2.50	2.36	3.57	2.29	1.64	3.72	2.96	1.67	2.10	2.87	3.82

Table 1 compares all the techniques for the four datasets in terms of the normalised squared error (*NSE*). For the GENOMICS and AROMA datasets we see that the LL and dAg approaches are better. LL continues its good behaviour for CARS and UJIndoorLoc, being clearly the best one (or close to the best one when it is the second), however a very different picture happens for the disaggregation strategy, which shows a quite poor performance on these datamarts, especially on UJIndoorLoc.

In order to offer a more comprehensive perspective of the results, Table 2 rank all the strategies within each technique for each dataset individually. The lower the better on these ranks, being the ranks averaged when they tie.

Focusing on the ranking results, for the AROMA and CARS datasets we see that the LL approach is better, obtaining a very good ranking when the KNN technique is used. Moreover, LL obtains the second best rank in the other datasets. SL obtains the best rank in UJIndoorLoc dataset. However, it shows a quite poor performance for GENOMICS and AROMA. dAg obtains good results in GENOMICS (the best) and AROMA, however its rank is not so good for the other datasets.

Finally, as mentioned in Sect.2, there exist many operating contexts where several dimensions could be disaggregated at the same time. Instead of just comparing the results when only one dimension is disaggregated, we could have taken into account all the possible disaggregating operations for that context and averaged the results (as an ensemble) for comparing them to the SL and LL approaches. This option, called dAg^{avg}, was analysed experimentally and showed worse results in general. Table 3 shows the overall rankings for each strategy and for each dataset when all the possible dimensions in each cube have been taken into account by means of averaging their predictions. The overall rankings for the dAg^{avg} methodology are always worse than the ones shown in Table 2 except for the CARS dataset whereas in UJIndoorLoc dataset, dAg and dAg^{avg} practically obtain the same rank.

Table 3. Summary results for all the strategies and for each dataset when all the disaggregation operations for each cube have been performed and their results have been averaged.

	SL	LL	dAg^{avg}	SLc
GENOMICS	3.70	1.93	2.42	4.37
AROMA	3.60	1.60	3.10	4.39
CARS	2.12	1.40	3.63	3.54
UJIndoorLoc	1.49	1.84	2.89	4.12

5 Related Work

As we mentioned in the introduction, the efforts for a full integration of data mining and OLAP tools have not been as common as originally expected. There are, though, some significant contributions for descriptive models. For instance, multidimensional association rules were firstly introduced in [17] and, since then, some related approaches have appeared in areas such as hierarchical association rules, subgroup discovery, granular computing [18] and others [7].

'Prediction cubes' [7,8], despite the term 'predictive' in their name, actually perform subgroup discovery or exploratory mining [23], where we want to have a metric (e.g., predictive accuracy) for a model on a given subset of the data (a cell in a cube) and see whether some cells have different metrics than others (hence being special). It is important to note that "Prediction Cubes" are not meant to aggregate outputs. They are not actually used to make predictions at several resolution levels of values that are unknown. In fact, they always work with a labelled test set to which they compare to get the metrics.

When looking at predictive modelling, the usual approach in the literature has been the *same level* approach (i.e., generating a view for the resolutions at hand). There is no versatile model that can work for the whole hierarchy in every dimension. A significant exception is the area of multilevel modelling (MLM) [4,12], also known as hierarchical (linear) modelling (HLM) [24], among

other names. This is an extension of linear, and non-linear, models such that the variables are measured at different levels of a global, usually linear, hierarchy. The first and key difference between a multilevel modelling problem and a multidimensional problem is hence that in the latter all the measurements take place at the lowest level (e.g., they come from facts in a multidimensional data warehouse). However, in multilevel modelling, measurements may take place at any level. As a result, in multilevel modelling, putting all the variables at the lowest level does not make sense, as it means that some of the input variables would have to be disaggregated (or repeated). The second difference is that in multilevel modelling hierarchies apply to *all* attributes. In other words, there is an orthogonal hierarchy, which can be applied to each attribute, depending on the level at which the value has been measured. So it is not actually applicable to a multidimensional database, where each attribute can be aggregated independently. The third difference is that in multilevel modelling the predictions are still made generally at the lowest level. In a multidimensional setting we want predictions at whatever level of aggregation. In addition, multilevel modelling has usually been addressed by linear (and occasionally non-linear) regression models with several assumptions about normality, homoscedasticity, independence, etc. Despite the differences, in [19], multilevel models are applied to a datamart. However, we still see a separate concept hierarchy that is applied to all dimensions, instead of having a particular hierarchy for each dimension, as usual in datamarts and OLAP tools, so it is not actually a multidimensional database.

Some connections have also been found with the work of Perlich and Provost in [21], where the authors introduce new aggregates that capture more information about the distributions. That is, instead of using simpler aggregates such as the MEAN, MODE or SUM, this work presents novel aggregates that empirically improve predictive modelling in high-dimensional (categorical) domains. The setting of hierarchies is different to our multidimensional setting, and our approach considers a given hierarchy where the natural aggregates in our case are MEAN and SUM. Nonetheless, it could be worth studying where more aggregates or statistics about the distributions (e.g., SKEW, VAR) could help to correct the MEAN and SUM aggregates.

About the disaggregation approaches, the dAg and SLc models were inspired by [10]. Actually, they do not disaggregate as it can be understood in a cube-space data mining [23], but this work made us realise that the disaggregation approach is not properly covered within the OLAP-style multidimensional data analysis. Our proposed SLc model is not equivalent to the approach presented in [10], since in their work authors needed the *exact*, or at least *accurate enough*, values for the aggregated target.

As a result, the problem of having several hierarchies, one for each dimension and seeing the problem (including predictions) at any possible resolution, is new. Also there is no general approach about how to apply any data mining technique to this kind of problem (and not only linear regression models or non-linear

variants). So, the multidimensional approach presented in this paper is more general in at least these two aspects.

6 Conclusions and Future Work

Multidimensional data is a rich and complex scenario where the same task can change significantly depending on the level of aggregation over some of the dimensions. This is the 'multidimensional context'. The approaches we have analysed are very general, and applicable to any set of off-the-shelf machine learning techniques. Three of the approaches can be considered as retraining approaches, whereas the LL approach the only reframing approach. From this distinction, we see that resources are an important criterion, as retraining a model again and again may become infeasible for some applications, and reframing a single, versatile model may be a much better option in cost-effective terms. Also the results are generally better for the LL approach. It may be the case that there are some criteria to choose the best option at each granularity. From our analysis, however, we have not found any clear pattern to make a different take-away recommendation other than the LL approach. In order to facilitate repeatability of the experiments, the software associated to this work is available at http://users.dsic.upv.es/~admarus/sw.html.

This work suggests many avenues for future work. One area we are undertaking is a modification of the LL approach where the aggregation function is substituted by a quantification procedure [2,11]. As quantification is able to correct some aggregation problems, we hope some quantification techniques (especially those for regression using crisp regression models [3] or soft regression models [15]) to be beneficial for the LL approach. The set of predictions from different approaches could also be used as an ensemble, hopefully leading to better results. Another further improvement could be strengthen the existing relationship among the different approaches by means of more experimentation, reinforcing in this way our knowledge about dependencies in predictions, in which particular circumstances any of the approaches is better or the different efficiencies for each methodology, which is critical in OLAP scenarios. Finally, even if the approaches analysed in this paper are general to work for any off-the-shelf machine learning techniques, there may be room for improvement if specific techniques are developed for the multidimensional setting: multidimensional KNN, multidimensional decision trees and multidimensional Naive Bayes.

Acknowledgments. This work was supported by the Spanish MINECO under grants TIN 2010-21062-C02-02 and TIN 2013-45732-C4-1-P, and the REFRAME project, granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences Technologies ERA-Net (CHIST-ERA), and funded by MINECO in Spain (PCIN-2013-037) and by Generalitat Valenciana PROMETEOII2015/013.

References

1. Agrawal, R., Gupta, A., Sarawagi, S.: Modeling multidimensional databases. In: Proceedings of the Thirteenth International Conference on Data Engineering, ICDE 1997, pp. 232–243. IEEE Computer Society (1997)
2. Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.: Quantification via probability estimators. In: IEEE ICDM, pp. 737–742 (2010)
3. Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Aggregative quantification for regression. DMKD **28**(2), 475–518 (2014)
4. Bickel, R.: Multilevel analysis for applied research: It's just regression! Guilford Press (2012)
5. Cabibbo, L., Torlone, R.: A logical approach to multidimensional databases. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, p. 183. Springer, Heidelberg (1998)
6. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. ACM Sigmod Record **26**(1), 65–74 (1997)
7. Chen, B.C.: Cube-Space Data Mining. ProQuest (2008)
8. Chen, B.C., Chen, L., Lin, Y., Ramakrishnan, R.: Prediction cubes. In: Proc. of the 31st Intl. Conf. on Very Large Data Bases, pp. 982–993 (2005)
9. Datahub: Car fuel consumptions and emissions 2000–2013 (2013). http://datahub.io/dataset/car-fuel-consumptions-and-emissions
10. Dhurandhar, A.: Using coarse information for real valued prediction. Data Mining and Knowledge Discovery **27**(2), 167–192 (2013)
11. Forman, G.: Quantifying counts and costs via classification. Data Min. Knowl. Discov. **17**(2), 164–206 (2008)
12. Goldstein, H.: Multilevel Statistical Models, vol. 922. John Wiley & Sons (2011)
13. Golfarelli, M., Maio, D., Rizzi, S.: The dimensional fact model: a conceptual model for data warehouses. Intl. J. of Coop. Information Systems **7**, 215–247 (1998)
14. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. SIGKDD Explor. **11**(1), 10–18 (2009)
15. Hernández-Orallo, J.: Probabilistic reframing for cost-sensitive regression. ACM Transactions on Knowledge Discovery from Data **8**(3) (2014)
16. IBM Corporation: Introduction to Aroma and SQL (2006). http://www.ibm.com/developerworks/data/tutorials/dm0607cao/dm0607cao.html
17. Kamber, M., Jenny, J.H., Chiang, Y., Han, J., Chiang, J.Y.: Metarule-guided mining of multi-dimensional association rules using data cubes. In: KDD, pp. 207–210 (1997)
18. Lin, T., Yao, Y., Zadeh, L.: Data Mining, Rough Sets and Granular Computing. Studies in Fuzziness and Soft Computing. Physica-Verlag HD (2002)
19. Páircéir, R., McClean, S., Scotney, B.: Discovery of multi-level rules and exceptions from a distributed database. In: Proc. of the 6th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining, pp. 523–532. ACM (2000)
20. Pastor, O., Casamayor, J.C., Celma, M., Mota, L., Pastor, M.A., Levin, A.M.: Conceptual Modeling of Human Genome: Integration Challenges. In: Düsterhöft, A., Klettke, M., Schewe, K.-D. (eds.) Conceptual Modelling and Its Theoretical Foundations. LNCS, vol. 7260, pp. 231–250. Springer, Heidelberg (2012)
21. Perlich, C., Provost, F.: Distribution-based aggregation for relational learning with identifier attributes. Machine Learning **62**(1–2), 65–105 (2006)
22. Team, R., et al.: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2012)

23. Ramakrishnan, R., Chen, B.C.: Exploratory mining in cube space. Data Mining and Knowledge Discovery **15**(1), 29–54 (2007)
24. Raudenbush, S.W., Bryk, A.S.: Hierarchical linear models: applications and data analysis methods, vol. 1. Sage (2002)
25. UCI Repository: UJIIndoorLoc data set (2014). http://archive.ics.uci.edu/ml/datasets/UJIIndoorLoc
26. Vassiliadis, P.: Modeling multidimensional databases, cubes and cube operations. In: Proc. of the 10th SSDBM Conference, pp. 53–62 (1998)