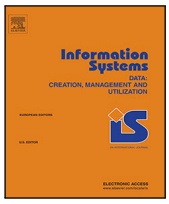
Information Systems 136 (2026) 102628

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is



Predicting multidimensional cubes through intentional analytics

Matteo Francia [a], Stefano Rizzi [a,*], Matteo Golfarelli [a], Patrick Marcel [b]

[a] DISI, University of Bologna, Italy
[b] LIFO, University of Orléans, France

ARTICLE INFO

Keywords:
Data cube
OLAP
Intentional analytics model
Regression

ABSTRACT

In an attempt to streamline exploratory data analysis of multidimensional cubes, the Intentional Analytics Model ha been proposed as a way to unite OLAP and analytics by allowing users to indicate their analysis intentions and returning cubes enhanced with models. Five intention operators were envisioned to this end; in this work we focus on the predict operator, whose goal is to estimate the missing values of a cube measure starting from known values of the same measure or other measures using different regression models. Although prediction tasks such as forecasting and imputation are routinary for analysts, the added value of our approach is (i) to encapsulate them in a declarative, concise, natural language-like syntax; (ii) to automate the selection of the best measures to be used and the computation of the models, and (iii) to automate the evaluation of the interest of the models computed. First we propose a syntax and a semantics for predict and discuss how enhanced cubes are built by (i) predicting the missing values for a measure based on the available information via one or more models and (ii) highlighting the most interesting prediction. Then we test the operator implementation, proving that its performance is in line with the interactivity requirement of OLAP session and that accurate predictions can be returned.

1. Introduction

In the past ten years, there has been a lot of interest in exploratory data analysis, the notoriously time-consuming process of interactively studying datasets to obtain insights [1]. Imagine having to investigate a multidimensional dataset as a data enthusiast with basic SQL and programming skills. Generating meaningful insights necessitates sending several queries for data profiling, hypothesis formulation, comparison calculations, etc., in addition to doing numerous statistical tests to guarantee the significance of insights. Note that these activities are still complex and tedious even for an expert in machine learning and OLAP, since they require selecting and trying different algorithms, tuning them, and quantifying the interest of findings once they have been obtained [2–5]. Despite the advancements in AI and LLMs, it is also recognized that applications such as Data Scientist, Analyst, and GTP-4o can, at the state of the art, support data scientists but not automatically pilot the extraction of insights [6,7]. Indeed, due to hallucinations, quantitative errors, and to the need for continuous fact-checking, their adoption as reliable end-to-end tools for exploratory data analysis is still limited [8].

In an attempt to effectively support exploratory data analysis of multidimensional cubes, the *Intentional Analytics Model* (IAM) has been proposed as a way to enhance traditional OLAP by coupling it with analytics [9]. The IAM approach relies on two pillars: (i) users explore a cube by expressing their analysis *intentions* rather than by asking queries and (ii) in return they receive, besides multidimensional data, knowledge insights in the form of models. To achieve (i), five intention operators were proposed, namely, describe (describes one or more cube measures at some aggregation level), assess (judges one or more cube measures with reference to some benchmark), explain (reveals the reason behind the values of a measure, for instance by correlating it with other measures), predict (shows data not in the original cube, derived for instance with regression), and suggest (shows data similar to those the current user, or similar users, have been interested in). Remarkably, all these operators share a natural language-like syntax aimed at hiding most technical details, so as to make the complexity of explorative data analysis transparent to inexperienced users while still allowing skilled users to steer the analysis process. As to (ii), IAM operates on *enhanced cubes*, i.e., multidimensional cubes coupled with *highlights* consisting of most interesting components of models automatically extracted from cubes; the interest of a component is computed based, for instance, on how novel and surprising it is to the user, or on how well it can replicate the values of a measure.

Among the five intention operators, describe, assess, and explain have been investigated in previous papers [10–12]. In this paper we

* Corresponding author.
E-mail addresses: m.francia@unibo.it (M. Francia), stefano.rizzi@unibo.it (S. Rizzi), matteo.golfarelli@unibo.it (M. Golfarelli), patrick.marcel@univ-orleans.fr (P. Marcel).

https://doi.org/10.1016/j.is.2025.102628
Received 31 March 2025; Received in revised form 26 August 2025; Accepted 11 September 2025
Available online 17 September 2025

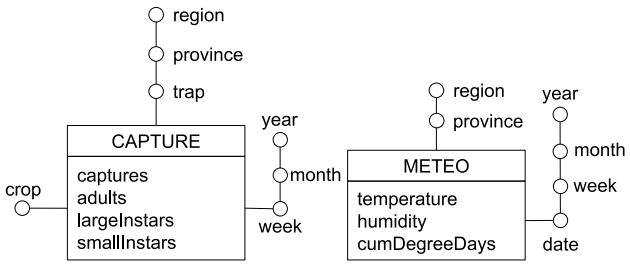


Fig. 1. Conceptual schemata for the CAPTURE and METEO cubes (for simplicity, each week is conventionally associated with a single month).

enrich the IAM picture by focusing on the predict operator. The goal of prediction is to make informed guesses about unknown outcomes or events using data, models, and algorithms. In the IAM scenario, this means estimating missing values of a cube measure m (called *target measure*) starting from known values of the same measure or other measures (*inspected measures*), possibly also taking into account one or more related cubes.

Example 1. Let a CAPTURE cube be given, whose multidimensional schema is shown in Fig. 1. This cube describes the captures of the brown marmorated stink bug (*Halyomorpha halys*), one of the main insect pest species causing economic damages to agricultural assets, in different weeks, traps, and crops. Weeks can be aggregated into months and years; traps can be aggregated based on the province and region where they are located. Captures are characterized by the age of the insects, so the cube contains four measures: the amount of captured adults, large instars, and small instars as well as the total captures (which should correspond to the sum of the other measures). On September 1st 2024, within the IAM framework, the user formulates the following intention based on the predict operator:

```
with CAPTURE predict adults by week, province
    for month between 'May 2024' and 'September 2024'
    using univariateTS
```

where adults is the target measure and all the other measures of CAPTURE are inspected to be used for prediction. Note that, since the temporal horizon of the intention spans four weeks in the future, this is a case of forecasting (the between predicate includes the extremes). Besides, the user requires to use a specific model type (univariate time series) to build the prediction. Fig. 2 sketches the IAM approach and the enhanced cube resulting from this intention. In the enhanced cube, the cube data are coupled with a model that encodes a prediction of the number of insects captured by traps scattered in the territory. The model includes two components for two different provinces; of these, Bologna (BO) is the highlight, i.e., the one yielding maximum interest. Intuitively, the interest of a component measures to what extent a variation in the target measure (adults in this case) is predictable from the inspected measures via that component.

Note that the using clause is the key to specify the models to be computed. While in Example 1 the user required to compute only one specific model (a univariate time series), (s)he could have as well required models of multiple types by listing them in the using clause, or even require all model types by omitting the clause (see Section 4.1 for the complete syntax of the predict operator). Remarkably, having different models automatically computed and evaluated in terms of their interest relieves the user from the time-wasting effort of coding and trying different possibilities, with intentions being up to 99% characters shorter than the necessary implementation of the operator (as discussed in Section 5.2).

Typical use cases of prediction are:

- *Forecasting*, where future values of m are predicted based on what happened in the past; for instance, the sales of a given product for the next days can be predicted based on the sales made in the past by also considering seasonality.
- *Imputation*, which replaces missing values of m in the past with estimated values based on the available information; for instance, the data stream collected from some sensor could have some interruptions due to a failure, and the missing values could be predicted based on those of a nearby sensor.
- *Allocation*, where a value of m at some given aggregation level is allotted at a finer aggregation level; for instance, a budget prevision defined for a product category could be allocated to the single products of that category based on the sales actually made last year.

Different types of regression models can be used to this end:

- *regression trees*, hierarchical and interpretable models where each internal node represents a test on a cube level or inspected measure, each branch represents the outcome of the test, and each leaf is the value predicted for the target measure [13];
- *random forests*, sets of regression trees trained on different parts of the same training set, whose predictions are obtained by averaging that of the single regression trees — making them less interpretable than those of regression trees [14];
- *models for time series analysis* (briefly, from now, *time series*), such as those computed by SARIMAX [15] and VARMAX [16], i.e., models for analyzing temporal sequences of data in order to make predictions based on recurrent and seasonal patterns (e.g., by summing up the effects of autoregression and moving average).
- *neural networks*, networks of artificial neurons linked with connections having different weights; once an input is supplied to a neural network, the prediction (output) is made by activating chains of neurons.

Among the different types, in this paper we restrict to regression trees, random forests, and time series. Regression trees and random forests are chosen since they return useful information about the importance of the features used to make the predictions. Time series are chosen since they are specific for learning autoregression and seasonal patterns. Neural networks are not considered to avoid using fully black-box models. As to use cases, in this paper we cover forecasting and imputation, while allocation is left for future work. Noticeably, all three model types we selected can be used for both forecasting and imputation.

Example 2. The result of the intention in Example 1 is computed as follows. First, the subset of facts since May 2024 (for clause) are selected from the CAPTURE cube (with clause) and aggregated by the province where the trap is located (by clause). In OLAP terms, a slice-and-dice and a roll-up operator are applied. Then, as required by the user's intention, a univariate times series model with one component for each province is computed. Finally, a measure of interest that expresses how well the values of adults are predicted by each component (i.e., for each province) is computed for all the components obtained, and the one with the highest interest (the highlight, i.e., the component yielding the most accurate prediction) is shown to the user.

We remark that, although tasks such as forecasting and imputation are routinary for expert analysts, the added value of our approach is

1. to encapsulate them in a declarative, concise, natural language-like syntax that can be easily understood by non-expert analysts as well;
2. to allow different degrees of specification for intentions, so as to make the operator helpful for a broad range of users: inexperienced users, who can just declare their goals with no worry for

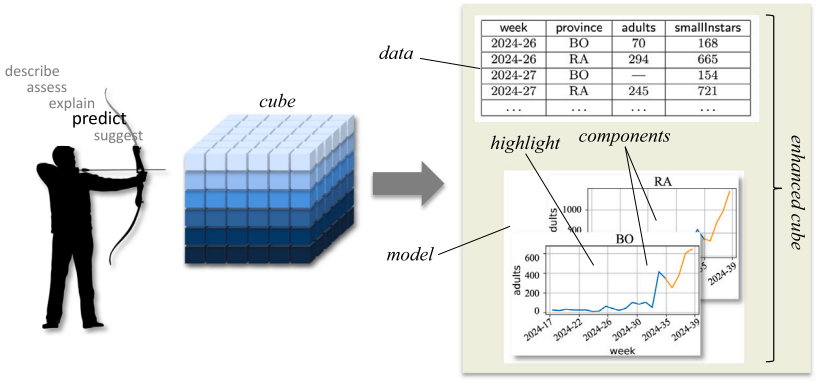


Fig. 2. The IAM approach.

technical details, and skilled users, who can steer the analysis process by specifying the model types and measures to be used for their predictions;
3. to automate the selection of the applicable model types and of the best measures to be used, the pre-processing phase, and the computation of models;
4. to automate the evaluation of the interest of the models computed, where the interest of a component essentially measures how well it can predict the values of the target measure;
5. to put together the cube data, the related models, and the highlight into a single artifact, the enhanced cube.

The remainder of this paper is structured as follows. In Section 2, after formalizing multidimensional cubes, we give a new definition of cube queries to (i) create future facts so as to transparently cope with forecasting scenarios, and (ii) enable the join between related cubes. In Section 3 we formalize prediction models and their components in compliance with the IAM framework, and explain how the interest of components is evaluated. In Section 4 we define the predict operator in terms of syntax and semantics, with specific reference to each model type considered. In Section 5 we present the results of a comprehensive set of experimental tests aimed to evaluate our approach from the points of view of time efficiency, writing complexity, and effectiveness. The paper is completed by Section 6, which discusses the related literature, and Section 7, where we draw the conclusions.

2. Formalities

To simplify the formalization and without loss of generality, we will restrict to consider linear hierarchies.[1]

Definition 1 (*Hierarchy and Cube Schema*)**.** A *hierarchy* is a triple $h = (L_h, \succeq_h, \geq_h)$ where:

(i) L_h is a set of categorical *levels*, each coupled with a *domain* $Dom(l)$ including a set of *members*;
(ii) $\succeq_h$ is a *roll-up* total order of L_h; and
(iii) $\geq_h$ is a *part-of* partial order of $\bigcup_{l \in L_h} Dom(l)$.

The top level of $\succeq_h$ is called *dimension*, the bottom level is denoted all_h and has a single member ALL_h. The part-of partial order is such that, for each couple of levels l and l' such that $l \succeq_h l'$, for each member $u \in Dom(l)$ there is exactly one member $u' \in Dom(l')$ such that $u \geq_h u'$.

Note that, since forecasting is a frequent kind of prediction, we will conveniently assume that the domain of temporal levels also includes members representing a set of future dates. Let l be a temporal level and t a time (for instance, t = NOW); we will denote with $Dom_t^-(l)$ and $Dom_t^+(l)$, respectively, the disjoint subsets of $Dom(l)$ preceding and following (or equal to) t ($Dom_t^-(l) \cup Dom_t^+(l) = Dom(l)$).

Definition 2 (*Cube schema*)**.** A *cube schema* is a couple $C = (H, M)$ where (i) H is a set of hierarchies; (ii) M is a set of numerical measures, where each measure $m \in M$ is coupled with one aggregation operator $op(m) \in \{\mathtt{SUM}, \mathtt{AVG}, \ldots\}$.

Example 3. For our working example we will use the CAPTURE and METEO cubes, whose conceptual schemata are depicted in Fig. 1 using the DFM [17]. Formally, it is CAPTURE = (H, M) with

$H = \{h_{\mathsf{Time}}, h_{\mathsf{Trap}}, h_{\mathsf{Crop}}\}$;
$M = \{\mathsf{captures}, \mathsf{adults}, \mathsf{largeInstars}, \mathsf{smallInstars}\}$;
$\mathsf{week} \geq_{\mathsf{Time}} \mathsf{month} \geq_{\mathsf{Time}} \mathsf{year} \geq_{\mathsf{Time}} \mathsf{all}_{\mathsf{Time}}$;
$\mathsf{trap} \geq_{\mathsf{Trap}} \mathsf{province} \geq_{\mathsf{Trap}} \mathsf{region} \geq_{\mathsf{Trap}} \mathsf{all}_{\mathsf{Trap}}$;
$\mathsf{crop} \geq_{\mathsf{Crop}} \mathsf{all}_{\mathsf{Crop}}$

and $op(\mathsf{captures}) = op(\mathsf{adults}) = op(\mathsf{largeInstars}) = op(\mathsf{smallInstars}) = \mathtt{SUM}$. In the part-of order of the Time hierarchy it is, for instance, 2024-26 $\geq_{\mathsf{Time}}$ June 2024 $\geq_{\mathsf{Time}}$ 2024 $\geq_{\mathsf{Time}}$ ALL. The METEO cube has three measures, namely temperature, humidity, and cumDegreeDays[2] and two dimensions, namely date and province, with $op(\mathsf{temperature}) = op(\mathsf{humidity}) = op(\mathsf{cumDegreeDays}) = \mathtt{AVG}$.

Aggregation is the basic mechanism to query cubes, and it is captured by the following definition of group-by set.

Definition 3 (*Group-by Set and Coordinate*)**.** Given cube schema $C = (H, M)$, a *group-by set* of C is a set of levels, exactly one from each hierarchy of H. The partial order induced on the set of all group-by sets of C by the roll-up orders of the hierarchies in H, is denoted with $\succeq_H$. A *coordinate* of group-by set G is a tuple of members, one for each level of G. Given coordinate γ of group-by set G, another group-by set G' such that $G \succeq_H G'$, and the coordinate γ' of G' whose members are related to the corresponding members of γ in the part-of orders, we will say that γ *roll-ups* to γ'. Conventionally, each coordinate roll-ups to itself.

[1] The presence of branches and diamonds in the hierarchies only affects the definition of group-by sets and, consequently, the definition of roll-up partial order and the computation of cube queries; it has no impact within the scope of this paper since we focus on models that operate at a fixed group-by set, the one stated in each intention.

[2] The *cumulative degree day* is a measure of heating or cooling often used in agriculture. In our scenario, it is computed as $\sum_{date}(avg_{hour \in date}(T_{hour}) - \overline{T})$, where $\overline{T} = 12.2°$ is the minimum temperature at which brown marmorated stink bugs develop.

Example 4. The top and bottom group-by sets of CAPTURE are $G^{\top}$ = {week, trap, crop} and $G^{\bot}$ = {$\mathsf{all}_{\mathsf{Time}}$, $\mathsf{all}_{\mathsf{Trap}}$, $\mathsf{all}_{\mathsf{Crop}}$}. Two more group-by sets of CAPTURE are G^1 = {week, province, crop} and G^2 = {month, region, $\mathsf{all}_{\mathsf{Crop}}$}, where $G^1 \succeq_H G^2$. G^1 aggregates captures by week, province, and crop, G^2 by month and region. Example of coordinates of the two group-by sets are, respectively, $\gamma^1 = \langle$ 2024-26, BO, corn$\rangle$ and $\gamma^2 = \langle$June 2024, Emilia-Romagna, ALL$\rangle$, where γ_1 roll-ups to γ_2.

The instances of a cube schema are called cubes and are defined as follows.

Definition 4 (*Cube*)**.** A *cube* over C is a triple $C = (G_C, M_C, \omega_C)$ where (i) G_C is a group-by set of C; (ii) $M_C \subseteq M$; (iii) ω_C is a partial function[3] that maps the coordinates of G_C to a numerical value for each measure $m \in M_C$.

Each coordinate γ that participates in ω_C, with its associated tuple of measure values, is called a *fact* of C. With a slight abuse of notation, we will write $\gamma \in C$ to state that γ is a fact of C. The value taken by measure m in the fact corresponding to γ is denoted as $\gamma.m$. A cube $C^{\top}$ whose group-by set is the top of $\succeq_H$ (i.e., it is the finest group-by set of C) and such that $M_C = M$, is called a *base cube*.

The next step is to define cube queries. First of all we distinguish between *temporal* and *non-temporal* queries, where the former have a temporal level (date, week, month, etc.) in their aggregation while the latter have $\mathsf{all}_{\mathsf{Time}}$.[4] To cope with forecasting scenarios, where the temporal horizon of the user's intentions goes beyond the present day (or, more precisely, the time when the cube was lastly refreshed by ETL), temporal cube queries must return "future facts" (with all measures set to null) as well. To this end, we extend the definition of cube queries as follows, by introducing a parameter t to specify the moment in time that separates the past from the future. Most commonly t will be set to the present time, but it could also be set differently to allow accountability since, as shown by the definition, the query results depend on t. In case t is set to some moment in the past, all cube facts dated after t are replaced with null facts so they can be forecast and the predictions previously made can be reproduced.

Definition 5 (*Cube Query*)**.** A *query* over cube schema C is a triple $q = (G_q, P_q, M_q)$ where (i) G_q is a group-by set of H; (ii) P_q is a (possibly empty) set of selection predicates each expressed over one level of H using either a comparison operators (=, ≥, between, etc.) or the set inclusion operator (e.g., crop in {'corn', 'wheat'}); (iii) $M_q \subseteq M$. Query q is called *temporal* if G_q includes a temporal level l_{temp}, *non-temporal* otherwise.[5] Let $C^{\top}$ be a cube over C. If q is non-temporal, the result of applying q to $C^{\top}$ is a cube $C = q(C^{\top})$ such that (i) $G_C = G_q$; (ii) $M_C = M_q$; (iii) ω_C assigns to each coordinate $\gamma \in C$ satisfying the conjunction of the predicates in P_q and to each measure $m \in M_C$ the value computed by applying $op(m)$ to the values of m for all the coordinates of $C^{\top}$ that roll-up to γ. If q is temporal, let t be a member of the temporal level l_{temp}. We denote $F = \bigtimes_{l \in G_q \setminus \{l_{temp}\}} Dom(l) \times Dom_t^+(l_{temp})$. The result of applying q to $C^{\top}$ at time t is the same defined for non-temporal queries but (i) all the facts whose coordinates for which the member of l_{temp} belongs to $Dom_t^+(l_{temp})$ are removed, and (ii) new facts are added by assigning to each coordinate $\gamma \in F$ satisfying the conjunction of the predicates in P_q a null value for each measure in M_C.

[3] This function is partial not only due to cube sparseness, but also because C does not contain future facts, i.e., facts corresponding to the future dates included in the domain of temporal levels, $Dom_{\mathrm{NOW}}^+(l)$.

[4] The use we make of the term *temporal query* in this paper is not strictly the one made in the literature on temporal query languages and temporal databases.

[5] For simplicity, in this paper we restrict to consider the case where at most one temporal level is present in the query group-by set; an extension to support two or more temporal levels in the aggregation mainly requires some minor adjustments to the syntax of the predict operator (introduced in Section 4.1).

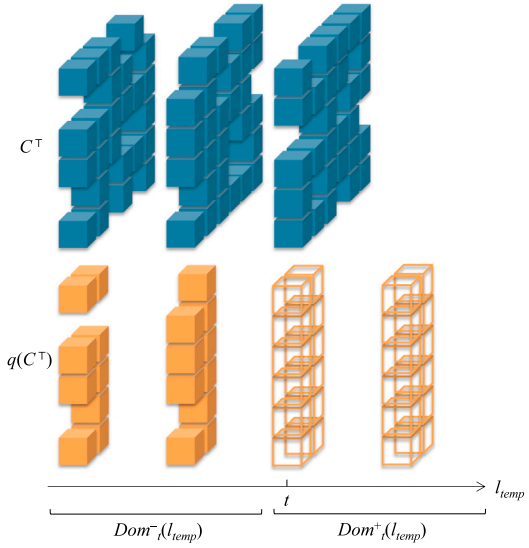


Fig. 3. Temporal cube queries: cube (top) and result of a temporal query q at time t (bottom). To compute the result of q (in orange), the real facts from t onwards are replaced with null facts. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Intuitively, as also shown in Fig. 3, if the predicate of q spans a time interval that follows t, the cube $C = q(C^{\top})$ resulting from the application of q to base cube $C^{\top}$ includes not only past facts (i.e., those selected and aggregated from $C^{\top}$), but also "future facts" corresponding to all the possible combinations of members of non-temporal levels of G_q and future times of temporal levels in G_q. By choosing t = NOW, this lets us transparently predict future facts in forecasting scenarios.

Example 5. The cube query over CAPTURE used in Example 1 is $q = (G_q, P_q, M_q)$ where G_q = {week, province, $\mathsf{all}_{\mathsf{Crop}}$}, P_q = {month between 'June 2024' and 'September 2024'}, and M_q = {adults}. Let $\mathsf{CAPTURES}_1$ be the resulting cube computed on August 30th, 2024; two coordinates of this cube are $\langle$2024-26, BO, ALL$\rangle$ with associated value 70 for adults (past fact) and $\langle$2024-39, RA, ALL$\rangle$ with associated null value for adults (future fact).

To let our predict operator use measures in one or more separate cubes to predict the target measure (e.g., to use temperature and humidity for predicting captures), so as to simulate the drill-across OLAP operator, we give a definition of cube *joinability*. Intuitively, two or more cubes are joinable if they share, either completely or partially, at least one hierarchy. A hierarchy is partially shared between two or more cubes if (i) at least one of its levels is shared and (ii) the roll-up orders among the shared levels are not conflicting.

Definition 6 (*Hierarchy Sharing*)**.** Let h_1, h_2 be two hierarchies in two different cubes. We denote $h_1 \simeq h_2$ if either $h_1 = h_2$ or

(i) $L = L_{h_1} \cap L_{h_2} \neq \emptyset$ and
(ii) $\forall l, l'$ s.t. $l \in L \wedge l' \in L,\ l \succeq_{h_1} l' \Leftrightarrow l \succeq_{h_2} l'$

In the first case, the shared hierarchy is obviously $h = h_1 = h_2$. In the second case, the shared hierarchy h includes all and only the levels in L with their roll-up order.

With a slight abuse of notation, we will overload the intersection operator in such a way that $\{h_1\} \cap \{h_2\} = \{h\}$ when $h_1 \simeq h_2$.

Definition 7 (*Joinability and Join*)**.** Let $C_1, \ldots, C_p$ be p cubes over cube schemata $C_1, \ldots, C_p$, respectively, be given, with $C_i = (H_i, M_i)$ (all M_i's

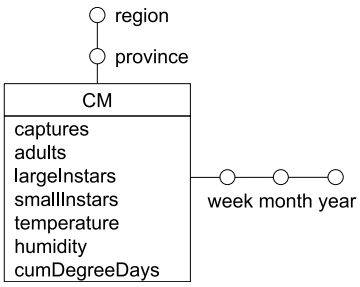


Fig. 4. Conceptual schema for the cube resulting from the join of CAPTURE and METEO.

are assumed to be pairwise disjoint). We say these cubes are *joinable* if $\bigcap_{i=1}^{p} H_i \neq \emptyset$. The cube resulting from the *join* between these cubes, $C_0 = \bigwedge_{i=1}^{p} C_i$, has schema

$$C_0 = (\bigcap_{i=1}^{p} H_i, \bigcup_{i=1}^{p} M_i)$$

Let $G^\top$ be the finest group-by set of C_0, and $q_i = (G^\top, TRUE, M_i)$ for $i = 1, \ldots, p$ be the queries that aggregate each cube C_i at $G^\top$. The coordinates of C_0 are the intersection of the coordinates of cubes $q_1(C_1), \ldots, q_v(C_p)$, i.e., the common coordinates of $C_1, \ldots, C_p$ aggregated at $G^\top$; each coordinate of C_0 is associated with all the measure values associated to the corresponding coordinates of the C_i's.

Intuitively, the join C_0 of two or more joinable cubes features the intersection of their hierarchies and the union of their measures, so (i) its group-by set is the finest common group-by set and (ii) its measures values, for each coordinate, are those of the corresponding coordinates in the joined cubes.

Example 6. Cubes CAPTURE and METEO partially share the temporal and the spatial hierarchies (see Fig. 1), hence, they are joinable; their join, CM = CAPTURE ∧ METEO, has measures captures, adults, …, humidity, cumDegreeDays (see Fig. 4). The finest group-by set of CM is $G^\top$ = {week, province}; an example coordinate of CM is $\gamma = \langle$2024-26, BO$\rangle$.

Let $C_0 = \bigwedge_{i=1}^{p} C_i$ be the join of p joinable cubes, and let C_0 be its cube schema. Then, any query $q = (G_q, P_q, M_q)$ over C_0 can be rewritten into p queries, $q_1, \ldots, q_p$, where $q_i = (G_q, P_q, M_q \cap M_p)$ is a query over the cube schema of C_i.[6] Consistent with Definition 5, the result $q(C_0)$ of applying q to C_0 is obtained by computing $q_i(C_i)$ for each i and then associating each coordinate of C_0 with the measure values associated to the corresponding coordinates of the C_i's.

3. Enhanced cubes

Models are concise, information-rich knowledge artifacts [18] that represent relationships hiding in the cube facts. A model is bound to (i.e., is computed over the levels/measures of) one cube, and is made of a set of components, each component being related to a subset of cube facts.

Definition 8 (*Model*)**.** A *model* is a tuple $\mathcal{M} = (type, alg, C, m, In, Out)$ where:

(i) $type$ is the model type;
(ii) alg is the algorithm used to compute Out;
(iii) C is the cube to which the model is bound (possibly resulting from a join);
(iv) m is the target measure of C;
(v) In is the set of r inspected measures of C supplied to alg to compute the model (note that In may include m or not);
(vi) Out is the set of model components, each associated with a tuple of hyper-parameter/value pairs to be fed to alg.

We distinguish two main categories of models: *time-aware* and *time-agnostic*.

- To compute a time-agnostic model $\mathcal{M}^{ag}$, alg is applied to C to predict all missing values of the target measure m; Out is made of a single component c allowing to predict all such values.
- A time-aware model $\mathcal{M}^{aw}$ requires that the group-by set G of C includes a level l from a temporal hierarchy, which we will call *temporal index*; thus, it can only be computed on the result of a temporal query. To compute $\mathcal{M}^{aw}$, the target and inspected measures are pivoted by the levels in $G\backslash\{l\}$ so as to generate a set of time series (one time series ts_{γ,m_i} for each coordinate $\gamma \in G\backslash\{l\}$ and for each measure $m_i \in \{m\} \cup In$); then, alg is applied to predict all missing values in the time series $ts_{\gamma,m}$ corresponding to m. Among time-aware models, we further distinguish:
  - *univariate models*, which include a component c_γ for each coordinate $\gamma \in G\backslash\{l\}$; c_γ corresponds to the prediction made for $ts_{\gamma,m}$ based on the ts_{γ,m_i}'s of the r inspected measures;
  - *multivariate models*, which include a single component c that predicts the missing values in all $ts_{\gamma,m}$ based on all the time series.

Note that, since in our approach a single measure m is predicted at a time, we do not use multivariate time-agnostic models.

Example 7. Let cube C include the weekly captures grouped by province (G = {week, province}). Let adults be the target measure and smallInstars be the inspected measure. Fig. 5 (top) shows two slices of C, those related to the provinces of BO (Bologna) and RA (Ravenna). The bottom part of the same figure shows the same data after pivoting adults and smallInstars by province, being week the temporal index; the result features four (vertical) time series, one for each province and each measure. In this case, a time-aware univariate model includes two components, one to predict the missing values in $ts_{\text{BO, adults}}$ based on $ts_{\text{BO, smallInstars}}$, and one to predict the missing values in $ts_{\text{RA, adults}}$ based on $ts_{\text{RA, smallInstars}}$. Conversely, a time-aware multivariate model includes a single component that predicts the missing values of adults based on *all* values available.

In the scope of this paper, we consider the following model types:

- *Regression tree*. In this case, the missing values of the target measure are predicted using a tree built via a tree regressor [13]. The models built can be either time-agnostic ($type$ = regressionTree) or time-aware univariate ($type$ = timeRegressionTree). The hyper-parameter space includes, for instance, maxDepth and minSampleLeaf.
- *Random forest*. Here, sets of trees are used for the prediction [14]. Like for regression trees, the models built can be either time-agnostic ($type$ = randomForest) or time-aware univariate ($type$ = timeRandomForest). The hyper-parameter space of random forests includes, for instance, maxDepth and minSampleLeaf of a single tree as well as the total number of trees (#estimators).
- *Time series*. Here, the missing values of the target measure are predicted using equations. The models built are time-aware and can be either univariate ($type$ = univariateTS, obtained for instance by applying the SARIMAX algorithm [15]) or multivariate

[6] This is easily proved since, by Example 6, G_q is common to all p cubes and each measure in C_0 belongs to one of those cubes.

week	province	adults	smallInstars
2024-26	BO	70	168
2024-26	RA	294	665
2024-27	BO	—	154
2024-27	RA	245	721
...	...	...	...

	adults		smallInstars	
week	BO	RA	BO	RA
2024-26	70	294	168	665
2024-27	—	245	154	721
...	...	...	...	...
	$ts_{\text{BO,adults}}$	$ts_{\text{RA,adults}}$	$ts_{\text{BO,smallInstars}}$	$ts_{\text{RA,smallInstars}}$

Fig. 5. Measure pivoting (week is the temporal index; the all_{Crop} column, whose values are set to ALL, is omitted for simplicity).

($type$ = multivariateTS, obtained for instance by applying the VARMAX algorithm [16]). The hyper-parameter spaces of SARIMAX and VARMAX include, for instance, the p and q orders for autoregressive and moving average modules.

All these model types require to train and optimize predictive machine learning algorithms. As normally done to this end, we consider 80% of the available (i.e., non-missing) data as the *training set* used to fit each algorithm, and the remaining 20% as the *test set* to verify the goodness of the prediction. If a temporal level is included in the group by set, we use it to sort the data and the test set consists of the last 20% of data; otherwise, the test set is uniformly randomly sampled.

The form taken by components depends on the model type as follows.

Definition 9 (*Component*)**.** For $type$ = regressionTree and $type$ = timeRegressionTree, a component is a couple $c_i = (\gamma_i, d_i)$ where γ_i is a coordinate and d_i is a tree. For $type$ = randomForest and $type$ = timeRandomForest, a component is a couple $c_i = (\gamma_i, D_i)$ where γ_i is a coordinate and D_i is a set of trees. For $type$ = univariateTS and $type$ = multivariateTS, a component is a couple $c_i = (\gamma_i, e_i)$ where γ_i is a coordinate and e_i is an equation that relates m to the inspected measures in In.

Some further explanations follow:

- For a time-agnostic regression tree there is exactly one component $c = (\gamma, d)$, in which γ is the (single) coordinate of $G^{\perp}$ and d is a regression tree where (i) each node is either an inspected measure or a level in G; (ii) each arc exiting a node is a Boolean predicate on that node; and (iii) each leaf is a value predicted for the target measure.
- For a time-aware regression tree, for each component $c_i = (\gamma_i, d_i)$, γ_i is a coordinate of $G \setminus \{l\}$ (l is the temporal index) and d_i is a regression tree where (i) each node is either an inspected measure or l; (ii) each arc exiting a node is a Boolean predicate on that node; (iii) each leaf is a value predicted for the target measure with reference to the slice γ_i.
- The same holds for random forests, except that each component includes a set of trees rather than a single tree.
- As to univariate time series, for each component $c_i = (\gamma_i, e_i)$, γ_i is a coordinate of $G \setminus \{l\}$ and e_i is an equation that relates m to the inspected measures with reference to slice γ_i. The form of the equation depends on the algorithm used (e.g., if the SARIMAX algorithm is used, the e_i has five parts: a seasonal one, an autoregressive one, an integrated one, a moving average one, and an exogenous one [15]).
- For a multivariate time series there is exactly one component $c = (\gamma, e)$, in which γ is the (single) coordinate of $G^{\perp}$ and e is an equation that relates m and the inspected measures. The form of the equation depends on the algorithm used. For instance, if the VARMAX algorithm is used, e_i has three parts: an autoregressive one, a moving average one, and an exogenous one [15]; differently from SARIMAX, VARMAX is multivariate since it handles vectors rather than single temporal values.

Note that using the SARIMAX and VARMAX algorithms requires to have the target measure among the inspected measures ($m \in In$).

As the last step in the IAM approach, cube C is enhanced by associating it with a set of models bound to C and with a *highlight*, i.e., with the most interesting model component(s):

Definition 10 (*Enhanced cube*)**.** An *enhanced cube* E is a triple of a cube C, a set of models $\{\mathcal{M}_1, \ldots, \mathcal{M}_z\}$ bound to C, and a highlight

$$\overline{c} = \{argmax_{\{c_i \in \bigcup_{j=1}^{z} Out_j\}}(interest(c_i))\}$$

We evaluate the interest of c_i, $interest(c_i)$, as the *coefficient of determination* R2 [19], which measures to what extent the variation in the target measure m is predictable from the inspected measures via that component. Specifically, R2 compares how well the component fits m in comparison with its average value; should the average value be a better prediction than the component, R2 is negative, in which case function $interest()$ returns 0, so the interest ranges in a continuous scale from 0 to 1. Note that the highlight is defined as a set since, in case of ties, it includes multiple components (all those with maximum interest). This perfectly fits the philosophy of our approach since it can be useful to let users gain a wider insight on the prediction. In terms of implementation, the model components are stored in a priority queue sorted by their interest; therefore, the highlight includes the first component(s) retrieved from this queue.

Example 8. Consider again the cube C shown at the top of Fig. 5. Here is a list of possible models bound to C:

- A time-agnostic regression tree bound to C is

  $\mathcal{M}_1 = (type = \text{regressionTree}, alg = \text{DecisionTreeRegressor},$
  $C, m = \text{adults}, In = \{\text{smallInstars}\},$
  $Out = \{(c, \text{maxDepth} = 6,$
  $\text{minSampleLeaf} = 10, \ldots)\})$

  where DecisionTreeRegressor is the algorithm implemented in the *scikit-learn* library,[7] $c = (\langle \text{ALL}, \text{ALL}, \text{ALL} \rangle, d)$, d is the tree shown in Fig. 6, and maxDepth and minSampleLeaf are among the hyper-parameters used to compute d. As previously mentioned, in this case inner nodes correspond to either measures (smallInstars) or levels (province or week).

[7] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html.

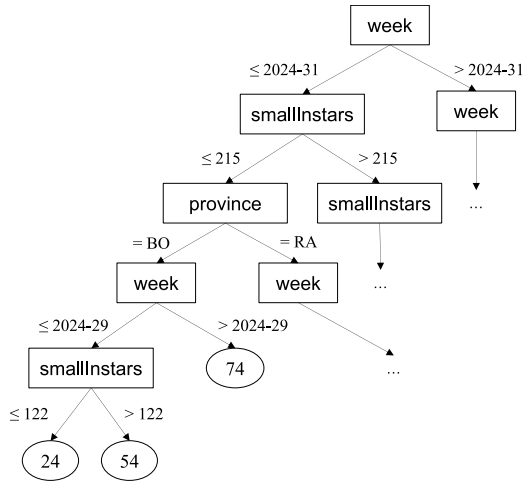


Fig. 6. An excerpt of the time-agnostic regression tree d for Example 8: boxes are either levels or inspected measures, arcs are predicates, while circles are the values predicted for the target measure.

Table 1
Interest computed for the components of the models in Example 8.

Model	Category	Model type	Component	Interest
$\mathcal{M}_1$	Time-agnostic	regressionTree	c	0.95
$\mathcal{M}_2$	Time-aware	timeRegressionTree	c^{tree}_{BO}	0.67
$\mathcal{M}_2$	Time-aware	timeRegressionTree	c^{tree}_{RA}	0.63
$\mathcal{M}_3$	Time-aware	univariateTS	c^{ts}_{BO}	0.98
$\mathcal{M}_3$	Time-aware	univariateTS	c^{ts}_{RA}	0.95

- On the other hand, a time-aware univariate regression tree bound to C, with temporal index week, is (hyper-parameters are omitted for simplicity)

$$\mathcal{M}_2 = (type = \text{timeRegressionTree},\ alg = \text{DecisionTreeRegressor},\ C, m = \text{adults},\ In = \{\text{smallInstars}\},\ Out = \{c^{tree}_{\text{BO}},\ c^{tree}_{\text{RA}}\})$$

where $c^{tree}_{\text{BO}} = (\langle \text{BO},\ \text{ALL}\rangle,\ d_{\text{BO}})$, $c^{tree}_{\text{RA}} = (\langle \text{RA},\ \text{ALL}\rangle,\ d_{\text{RA}})$, and d_{BO} is shown in Fig. 7. As previously mentioned, in this case inner nodes correspond to either measures (smallInstars) or to the temporal index (week).

- Finally, a univariate time series bound to C is

$$\mathcal{M}_3 = (type = \text{univariateTS},\ alg = \text{SARIMAX},\ C, m = \text{adults},\ In = \{\text{adults},\ \text{smallInstars}\},\ Out = \{c^{ts}_{\text{BO}},\ c^{ts}_{\text{RA}}\})$$

where $c^{ts}_{\text{BO}} = (\langle \text{BO},\ \text{ALL}\rangle,\ e_{\text{BO}})$ and $c^{ts}_{\text{RA}} = (\langle \text{RA},\ \text{ALL}\rangle,\ e_{\text{RA}})$. The equations are complex, for simplicity here we just show the autoregressive part of e_{BO}:

$$\text{adults}_t = \beta + \sum_{i=1}^{p}(\alpha_i \cdot \text{adults}_{t-i}) + \epsilon_t$$

where β is a constant value, α_i is a numeric coefficient by which we multiply the lagged data (how much the value at time $t-i$ affects the value at time t), lags are summed up to the maximum shift p, ϵ_t is called *residual* and represents the difference between our prediction for period t and its correct value. α and β are fitted by SARIMAX, while p is a hyper-parameter.

Table 1 summarizes the components with their interest; if all the models above were required by the user's intention, the highlight would turn out to be c_{BO} in $\mathcal{M}_3$. As normally done in predictive machine learning, we use 80% of the available data as the training set and the remaining 20% as the test set. Fig. 8 shows, for component c_{BO} of $\mathcal{M}_2$ and $\mathcal{M}_3$, the training and test sets and the prediction for the missing data.[8]

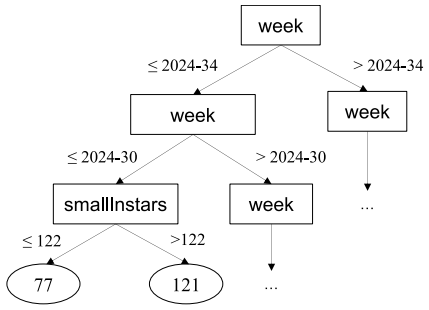


Fig. 7. An excerpt of the time-aware regression tree d_{BO} of BO for Example 8.

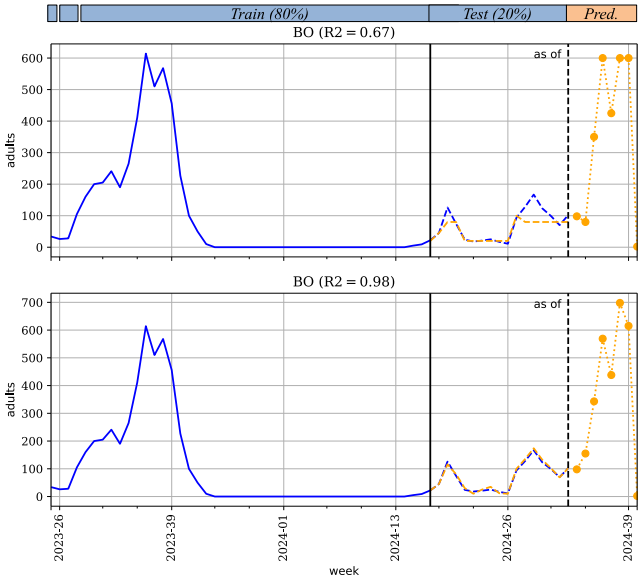


Fig. 8. Predictions made for BO via $\mathcal{M}_2$ (top) and $\mathcal{M}_3$ (bottom); the training sets are shown in solid lines, the test sets in dashed lines (real data in blue, predicted data in orange), and the model predictions for the missing data in dotted lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. The predict operator

The predict operator provides an answer to the user asking "how can I fill the missing values of this measure for my cube?". The cube is enhanced by computing models that predict the missing values (which may entail adding new cube slices for future data), and adding a highlight on the most interesting prediction. In the remainder of this section we present the operator syntax and its semantics; then we go into detail of how each model is trained and used to build a prediction.

[8] To make the chart better readable, we included only the last part of the training set. Finally, the enhanced cube E for this example includes cube C, the models $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$, and the highlight c_{BO}.

4.1. Syntax

As done for cube queries, we distinguish *temporal intentions* (based on temporal cube queries) from *non-temporal intentions*. Let us start with the latter.

Let $C_1^\top, \ldots, C_p^\top$ be p joinable cubes and $C = (H, M)$ be the schema of their join. The syntax for predict is (optional parts are in brackets):

with $C_1^\top[, \ldots, C_p^\top]$ predict m
  by $l_1, \ldots, l_n$ [for P]
  [from $m_1, \ldots, m_r$]
  [using $type_1, \ldots, type_z$]

where $m \in C$ is the target measure; P is a set of selection predicates, each expressed on one level of H; $\{l_1, \ldots, l_n\}$ is a group-by set of H; $m_1, \ldots, m_r$ are the inspected measures; $type_i \in$ {regressionTree, timeRegressionTree, randomForest, timeRandomForest, univariateTS, multivariateTS} is a model type. The different clauses take the following roles:

- The with clause specifies the cubes(s) on which the intention is executed.
- The target clause specifies the target measure.
- The by clause specifies how the cube(s) must be aggregated.
- The for clause specifies a selection on the cubes(s).
- The from clause specifies the inspected measures (possibly including also m).
- The using clause specifies which model types are to be computed.

Note that the members mentioned in a predicate on level l within the for clause must be part of $Dom(l)$.[9]

The syntax for a temporal intention is the same as above, except that the by clause must include one temporal level l_{temp} and that an optional clause may be added:

[as of t]

where t is a member of l_{temp}, to specify the time to be used for separating the future from the past. In a temporal intention, the for clause must include a predicate to delimit the members of l_{temp} (as done, for instance, in the intention of Example 1).

Partially-specified intentions are interpreted as follows:

- If the from clause is not specified, models are created for all measures in M including m.
- If the using clause is not specified, all model types are considered (except time-aware models in case of a non-temporal query).
- If the as of clause is not specified, t is set to the current time.

Example 9. Examples of predict intentions are, besides the one in Example 1,

with CAPTURE predict captures by province, crop
  for region = 'Emilia – Romagna'
  from captures, adults
with CAPTURE, METEO predict adults by month, region
  for month between 'May 2022' and 'September 2024'
  using regressionTree as of 'August 2024'

The first one is a non-temporal intention that imputes the missing values of captures for each province in Emilia-Romagna and each crop based on captures itself as well as on adults; since no temporal level is part of the by clause, only time-agnostic models can be computed. The second one is a temporal intention that forecasts the monthly future values of adults by region via time-agnostic regression trees, based on all measures available in the CAPTURE and METEO cubes; due to the as of clause, the forecast covers August and September 2024.

4.2. Semantics

In this section we define the operator semantics by explaining how an intention formulated at time t is computed:

1. If $p > 1$, i.e., two or more cubes are specified in the with clause, compute the cube C_0 resulting from their join.
2. Execute query $q = (G_q, P_q, M_q)$ over C_0, where $G_q = \{l_1, \ldots, l_n\}$, $P_q = P$, and $M_q = \{m, m_1, \ldots, m_r\}$. Let $C = q(C_0)$ be the cube resulting from the execution of q over C_0 at time t. Consistent with Definition 5, in case of a temporal intention this may imply the creation of new facts with null measure values.
3. For $1 \leq j \leq z$, compute model $\mathcal{M}_j = (type_j, alg_j, C, m, \{m_1, \ldots, m_r\}, Out_j)$; in case $\mathcal{M}_j$ is time-aware, this requires measures $m, m_1, \ldots, m_r$ to be preliminarily pivoted. Note that, in case new facts with null measure values have been created at the previous step, these are not used for training but only for storing the prediction.
4. For each $c \in Out_j$ compute $interest(c)$.
5. Find the highlight
$$\overline{c} = \{argmax_{c \in \bigcup_j Out_j}(interest(c))\}$$
6. Return the enhanced cube E consisting of C, $\{\mathcal{M}_1, \ldots, \mathcal{M}_z\}$, and $\overline{c}$.

The execution of the query (steps 1 and 2) is demanded to a relational-OLAP engine such as Oracle 19c; it requires translating the intention into an SQL statement in which the with clause determines the FROM clause, the for clause determines the conjunction of predicates in the WHERE clause, the by clause determines the GROUP BY clause, and the predict and using clauses determine the measures to be included in the SELECT clause together with the group by attributes.

Steps 3–6 are performed via a Python program. Specifically, at step 3, the result of the SQL query is used to compute the models via algorithms implemented using well-known libraries such as scikit-learn and statsmodels. The computation of $\mathcal{M}_j$ involves the following phases:

1. **Check applicability**: verify that the computation of $type_j$ on C_0 is feasible given the intention.
2. **Pre-process**: prepare the data in C_0 for the application of alg_j.
3. **Split training and test sets**. Firstly, we discard the facts having null values in the target measure (since they will be filled with the final prediction) and those having null values in the inspected measures (since they cannot be used for training the model). Then, we take 20% of the facts as a test set and the remaining 80% as the training set. Precisely, if the cube is the result of a temporal query, we take the last 20% of the data sorted by time [20]; otherwise, the test set is uniformly randomly sampled.
4. **Train and predict**. Given the hyper-parameter space for alg_j, we randomly check the performance of a fixed number of hyper-parameter configurations using 5-fold cross-validation. Given the best configuration, we apply alg_j to both the training and test set to predict the null values of the target measure.

Eventually, the Python program computes the interest (i.e., the coefficient of determination R2) for each component (step 4), selects the highlights with the highest interest (step 5), and returns the enhanced cube (step 6).

In Sections 4.3, 4.4, and 4.5 we specifically describe how phases 1 and 2 are carried out for each model type.

[9] Since, as mentioned in Section 2, the domain of temporal levels also includes future dates, this constraint does not prevent the predict operator to be used for forecasting in temporal intentions.

4.3. Regression trees

A tree regressor [13] is a machine-learning model that makes predictions using a tree model, where leaves represent predicted values and branches represent conjunctions of features that lead to those values. Regression trees are among the most popular machine learning algorithms given their intelligibility and simplicity.

Check applicability. The regression tree model can be applied to any intention, in both time-aware and time-agnostic modes. When the intention involves a temporal query, the regression tree predicts the values of each slice generated by pivoting the data by the temporal index.

Pre-process. Well-known implementations of regression trees require numeric input data. The levels' members are categorical (e.g., product or category names), thus they must undergo *one-hot encoding*, i.e., a pre-processing technique that creates one binary feature for each member. For each cube fact, the features corresponding to the categorical members of the fact are set to 1. Note that, since the levels' members are usually not ordinal, ordinal encoding cannot be applied.

4.4. Random forests

A random forest regressor [14] is an ensemble learning method for regression that leverages a multitude of regression trees. Random forests are not as intelligible as a simple regression tree, but they can solve some overfitting issues of simpler regression trees. Since random forests leverage regression trees, the phases are the same as in Section 4.3. Predictions are made by averaging the values returned by the individual trees.

4.5. Time series

With time series, in this paper, we refer to the machine learning models that can make predictions by considering patterns that are recurrent over time (e.g., at a seasonal or weekly scale). Although through feature pre-processing we can adapt regression trees and random forests to consider seasonal patterns (e.g., by injecting additional features with values shifted by time [20]), time series models consider these patterns by design.

Time series models can be distinguished into two classes [20], univariate and multivariate, depending on the measures involved in the prediction. Among the possible machine learning approaches belonging to each category, we use SARIMAX [15] (univariate) and VARMAX [16] (multivariate). Although more complex approaches and frameworks exist (e.g., AutoTS [21]), we select these two approaches because they can be more easily interpreted by end users.

4.5.1. Univariate time series

Univariate time series refer to machine learning models that predict *one* target measure using *one or more* inspected measures (possibly including the target measure itself, in which case the model is said to be *autoregressive*).

Check applicability. SARIMAX is a time-aware model applied only when the intention involves a temporal query (i.e., there is a temporal level in the by clause).

Pre-process. Given an intention that entails a temporal query, the cube is pivoted by the temporal index. This produces several time series (one for each combination of slices) of numeric values for the target measure, which do not require any encoding.

4.5.2. Multivariate time series

Multivariate time series are machine learning models that predict *many* target measures using *two or more* inspected measures (possibly including the target measures themselves). Multivariate time series have the same phases as univariate time series (see Section 4.5.1), except phase 1.

Check applicability. VARMAX is applied only when the intention involves a temporal query and at least two members of the temporal index are selected by the for clause. Additionally, VARMAX can be successfully applied only to *stationary* data (intuitively, time series that are stable over the long term); tests such as the Augmented Dickey Fuller one can determine whether a time series is stationary [20].

5. Experimental tests

The prototype we developed to test our approach uses the simple multidimensional engine described in [22], which in turn relies on the Oracle DBMS to execute queries on a star schema based on multidimensional metadata (in principle, the prototype could work on top of any other multidimensional engine). The algorithms are imported from the *scikit-learn* and *statsmodels* Python libraries. The number of hyperparameter configurations we test is 20 for each component. The code is publicly available at https://github.com/big-unibo/predict.

We remark again that the contribution of our approach is not focused on the single techniques used for prediction (other techniques could be easily plugged-in to build additional models), but on providing users with a declarative, concise, natural language-like syntax to express their prediction intentions, as well as to make transparent to users the selection and computation of the models and the interest-based ranking of the resulting insights. To the best of our knowledge, no other approaches have the same goal; for this reason, the only comparison we will make in this section is between the complexity of writing an intention using the predict operator and the complexity of writing the SQL and Python code necessary to compute the models (Section 5.2).

5.1. Effectiveness

The effectiveness of our approach is assessed by verifying the accuracy of the prediction, taking also in account the patterns that are known to be present in the data. To this end, we executed two groups of test.

In the first group, we evaluate the prediction accuracy on the real data stored in the CAPTURE cube. As sketched in Fig. 9, we set the as of clause in such a way that some actual data (in most cases, those of the last 20 weeks) are not taken into account to compute the prediction, so they can be used as a ground truth. The accuracy of a prediction is then measured, via the R2 coefficient [19], by comparing it with this ground truth; specifically, we compute the accuracy of each model as the average of the accuracy of its components (the standard deviation is always lower than 0.53). As shown in the following four subsections, the experiments are focused on progressively expanding in different ways the data used for the prediction. The using clause is always left unspecified, thus all applicable model types are computed; when the from clause is unspecified (e.g., in Section 5.1.2), all available measures are inspected. This also gives an idea of how the predict operator makes the underlying complexity transparent to inexperienced users, who may be unsure about which are the best model types and measures to be used for a prediction. The cubes used in the experiments represent trends of data over time, so we will mainly focus on temporal intentions. Since time-agnostic models are also computed in temporal queries, this does not prevent the tests from covering all the functionalities of the predict operator.

In the second group of test, discussed in Section 5.1.5, we use synthetic data were some patterns have been artificially injected and we check to what extent the predict operator can detect them. The intentions are written in such a way that 5% of data are not taken into account to compute the prediction and are used as a ground truth. Like for the first group, we measure accuracy via the R2 coefficient.

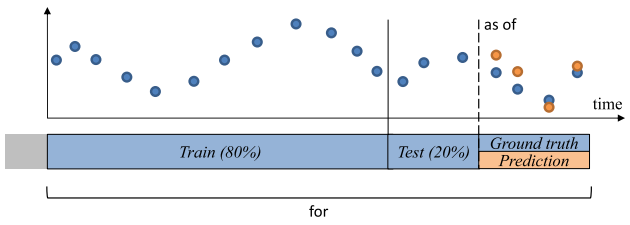


Fig. 9. Training and test sets and ground truth.

5.1.1. Adding inspected measures

In the CAPTURE cube, we know that measure captures should be the sum of adults, largeInstars, and smallInstars. We can expect that the predict operator learns this relationship and that the prediction of future adults improves its accuracy if adults, smallInstars, largeInstars, and captures are incrementally added into the from clause as inspected measures. The intentions used for this test are:

with CAPTURE predict adults by week
for month between 'May 2021' and 'September 2023' and province = 'BO'
as of '2023-20'...

$I1$:... from adults

$I2$:... from adults, smallInstars

$I3$:... from adults, smallInstars, largeInstars

$I4$:... from adults, smallInstars, largeInstars, captures

The prediction, as determined by the as of and for clauses, covers 20 weeks (5 months in 2023, corresponding to about 18% of the time span determined by the as of clause); the actual data of adults for these weeks are not considered to build the prediction and are used as the ground truth to assess its accuracy.

Fig. 10(a) shows, for each intention, the average accuracy of the components of each model, together with the accuracy of the highlight. It is apparent that predicting the captures of adults is hard when considering only its previous values (we recall that $|In|$, on the abscissa of the chart, is the number of inspected measures used to compute the model; thus, $|In| = 1$ corresponds to $I1$ where only adults is involved). Adding only smallInstars ($|In| = 2$, i.e., $I2$) adds noise to the prediction, reducing the prediction accuracy. The accuracy sensibly increases when measure largeInstars is added to the from clause ($|In| = 3$, i.e., $I3$), since largeInstars are more correlated to adults than smallInstars. When captures is added as well ($|In| = 4$, i.e., $I4$), all the models learn the inter-measure relationship providing predictions with more than 0.90 accuracy. Interestingly, univariateTS is the only model providing an accuracy of 0.99. Overall, we conclude that the predict operator is capable of properly capturing the existing (linear) relationship among the cube measures.

5.1.2. Increasing the time span

Here we evaluate how the accuracy of a prediction changes when progressively more past data are used to build it. The intention adopted to this end is the following:

with CAPTURE predict adults by week
as of '2023-34'
for province = 'BO'...

$I1$:... and month between 'January 2023' and 'September 2023'

$I2$:... and month between 'January 2022' and 'September 2023'

$I3$:... and month between 'January 2021'and 'September 2023'

where the as of clause fixes the length of the prediction and of the ground truth to cover the last 6 weeks (1.5 months in 2023). Differently from the previous intentions, here the ground truth covers only 6 weeks to leave enough data for training and testing within a single spring/summer.

Fig. 10(b) shows how the prediction average accuracy changes by increasing the time span from 1 to 3 years. The univariateTS model type is the only one capable of effectively learning the relationship between adults and capture for all time spans. The reason is that this model can mathematically express it in the form of an equation even when a single year of past data is considered; thus, it consistently generates accurate predictions across all time spans. Conversely, all other models rely on trees that encode recurrent patterns; hence, they benefit from larger datasets and improve their predictions as more historical data is included.

5.1.3. Adding slices

Since predictions are also made by capturing the relationships between different cube slices, in this section we evaluate how the accuracy changes when the provinces in Emilia-Romagna are progressively added to the data considered.

with CAPTURE predict adults by week from smallInstars, captures
as of '2023-20'
for month between 'May2021' and 'September2023'...

$I1$: ... and province in ('BO')

$I2$: ... and province in ('BO', 'RA')

$I3$: ... and province in ('BO', 'RA', 'FC')

where the as of clause fixes the length of the prediction and of the ground truth to 20 weeks.

Over the period being considered, Bologna (BO) and Ravenna (RA) had similar distributions of captures. In contrast, the captures in Forlì-Cesena (FC) were slightly different (fewer captures with some delayed spikes). This is confirmed in Fig. 10(c) by the predictions of the regressionTree, randomForest, and multivariateTS models ($|C|$ is the cardinality of the cube as determined by the for clause): their predictions are good when considering Bologna alone and when Ravenna is considered as well, and slightly worse when Forlì-Cesena is added. As to the timeRegressionTree and timeRandomForest models, their performance tends to decrease since the predictions made via these models for Ravenna and Forlì-Cesena are not accurate, hence, their inclusion reduces the average accuracy. Remarkably, multivariateTS achieves an accuracy of 0.98 when Bologna and Ravenna are considered and 0.96 when also Forlì-Cesena is added; this confirms that the similarity of capture distributions is effectively used by multivariateTS to make the prediction.

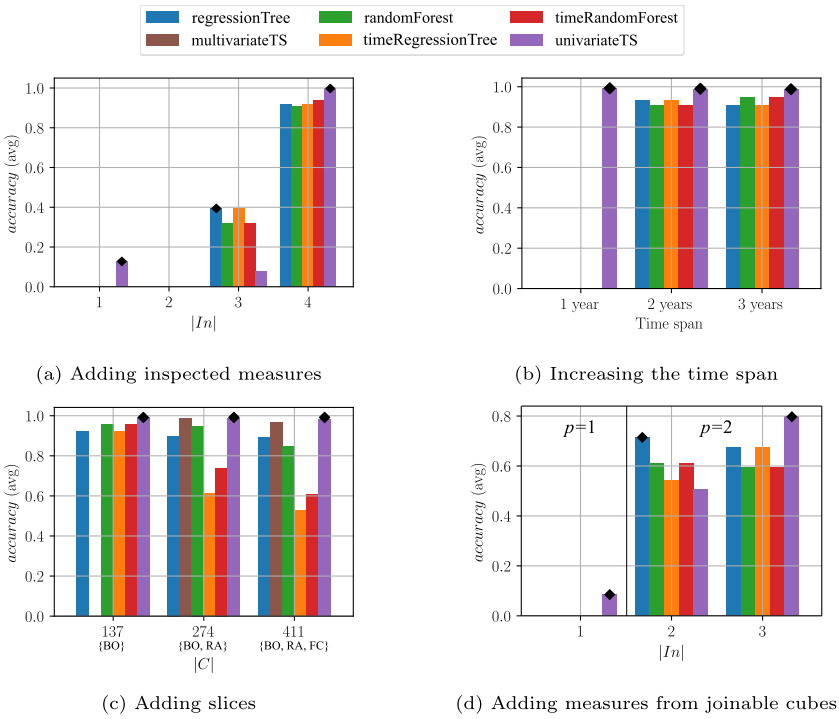
5.1.4. Adding joinable cubes

According to domain experts, the captures of adults should be affected by the temperature, specifically, by the so-called *cumulative degrees days*, i.e., the length of time during which temperatures are above a given threshold (in this case 12.2°, the temperature at which the brown marmorated stink bug starts to spread). As a consequence, we can expect that the prediction of adults will improve if the METEO cube, which stores the cumDegreeDays and temperature measures,[10] is added to the with clause:

with CAPTURE predict adults by week
as of '2023-20'
for month between 'May 2021' and 'September2023' andprovince = 'BO'

$I1$: ... from adults

[10] We could not use the humidity measure since it has too many missing values.



(a) Adding inspected measures (b) Increasing the time span

(c) Adding slices (d) Adding measures from joinable cubes

Fig. 10. Effectiveness tests; on the vertical axis, the average accuracy of the components of each model and the highlight accuracy (shown by a black diamond).

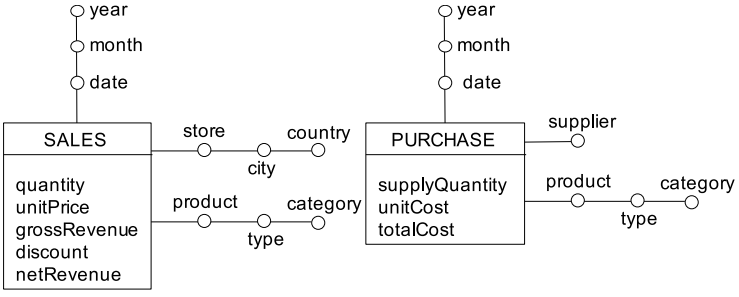


Fig. 11. Conceptual schemata for the SALES and PURCHASE cubes.

with CAPTURE, METEO predict adults by week

as of '2023-20'

for month between 'May2021' and 'September2023'and province = 'BO'

$I2$: ... for adults, cumDegreeDays

$I3$: ... for adults, cumDegreeDays, temperature

where the as of clause fixes the length of the prediction and of the ground truth to 20 weeks.

As shown in Fig. 10(d) (where p is the number of cubes in the with clause), when considering the CAPTURE cube only, the prediction accuracy is slightly above 0.1, while when joining it with the METEO cube it raises to 0.65 ($I2$) and 0.79 ($I3$). This suggests that the possibility of integrating additional cubes in the prediction is indeed an added value to the predict operator. The highest prediction accuracy is achieved by the regressionTree model for $I2$ and by the univariateTS model for $I3$. While adding the temperature measure affects the regressionTree model only slightly, the univariateTS model is able to exploit the periodic trend of temperature to improve the prediction.

5.1.5. Detecting patterns in synthetic data

To further assess the effectiveness of the predict operator, we consider two additional SALES and PURCHASE cubes [12], whose conceptual schemata are depicted in Fig. 11. These cubes store synthetic data, with three patterns artificially injected:

1. Measure discount is computed by applying to grossRevenue a percentage randomly chosen among 0%, 5%, and 10% (in the average, 5%).
2. Measure netRevenue is computed as grossRevenue − discount.
3. Measure unitCost is computed as unitPrice/2 plus a uniformly distributed random noise in $[-\frac{\text{unitPrice}}{10}, \frac{\text{unitPrice}}{10}]$ and moved backward by 30 days, to simulate that the fluctuations in the price of products follow the ones in their cost.

Then, we expressed the following intentions to verify that predict is capable of detecting these patterns:

$I1$: with SALES predict discount by date as of '2022-12-13'

$I2$: with SALES predict netRevenue by type

Table 2
Detecting patterns in synthetic data.

Intention	$I1$	$I2$	$I3$	$I4$
$\|C\|$	365	102	365	365
Time (s)	58.8	0.98	44.9	96.0
Accuracy (avg)	0.97	0.94	0.95	0.40
Accuracy (highlight)	0.98	0.95	0.98	0.40
♯ pred. values	18	6	18	18

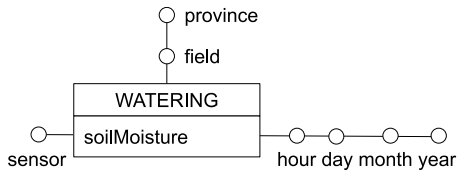


Fig. 12. Conceptual schema for the WATERING cube.

$I3$: with SALES predict netRevenue by date as of '2022-12-13'

$I4$: with SALES, PURCHASE predict unitPrice by date from unitCost as of '2022-12-13'

Note that $I1$, $I3$, and $I4$ are temporal intentions while $I2$ is not; besides, $I4$ requires to join the two cubes to predict unitPrice (from the SALES cube) based on unitCost (from the PURCHASE cube).

The results are summarized in Table 2 ($|C|$ is the cardinality of the cube, i.e., the number of facts, as determined by the for clause). The accuracy is above 0.9 for $I1$, $I2$, and $I3$, proving that the operator recognizes and correctly exploits the injected patterns. For $I4$ the accuracy is significantly lower. The reason for this is that our temporal models are autoregressive, thus, they compute the future values of unitPrice based on the past/current values of unitPrice itself and on the *current* value of unitCost; however, since the value of unitPrice on a given day is actually related to the value of unitCost 30 days before, this pattern is only partially captured by the operator. Note that the operator effectiveness for $I4$ is also affected by the noise introduced by aggregating and averaging unitPrice and unitCost over all products and all stores in the same date, thereby cumulating and propagating the noise at a coarser level of detail.

5.2. Efficiency

First of all, we measured the complexity (as the number of characters [23]) of writing predict intentions vs the underlying code. By averaging all the intentions in Section 5.1 and in the remainder of this section, it turns out that the average length of an intention is about 200 characters. Thus, our approach saves about 99% of complexity with respect to writing cube queries in SQL and writing the Python implementation necessary to compute the models (2000 characters for regressionTree and randomForest, 5500 for SARIMAX and VARMAX, 3400 for timeRegressionTree and timeRandomForest, and 3000 for managing and transforming query results and enhanced cubes).

As to time efficiency, since the CAPTURE cube has a small cardinality, here we use a different cube, WATERING [24], which stores the hourly soil moisture sampled by sensors (a regular grid of 12 sensors placed at different distances and depths from an irrigation dripper) located in different fields (see Fig. 12 for the conceptual schema). By doing so, we can also observe how the predict operator works with different datasets.

The efficiency of the predict operator is assessed on the WATERING cube by progressively increasing (i) the number of slices selected by the for clause and (ii) the time span of the prediction (both in terms of training and test data). The tests were run on an Intel(R) Core(TM)i7-6700 CPU@3.40 GHz CPU with 32 GB RAM; each intention was executed 10 times and the average results are reported.

5.2.1. Scaling up the slices

To assess how performances scale when inspected measures are incrementally considered by the operator, we run seven intentions that predict (using all the available model types) the hourly values of soilMoisture in a given field and during a time span of two days for an increasing number of sensors:

with watering predict soilMoisture by hour, sensor
for field = 'Field-1'
and hour between '2022-07-0110 : 00 : 00' and '2022-07-0310 : 00 : 00'...

$I1$:... and sensor in ('S20-0', 'S40-0')

$I2$:... and sensor in ('S20-0', 'S40-0', 'S60-0')

$I3$:... and sensor in ('S20-0', 'S40-0', 'S60-0', 'S20-25')

$I4$:... and sensor in ('S20-0', 'S40-0', 'S60-0', 'S20-25', 'S40-25')

$I5$:... and sensor in ('S20-0', 'S40-0', 'S60-0', 'S20-25', 'S40-25', 'S60-25')

$I6$:... and sensor in ('S20-0', 'S40-0', 'S60-0', 'S20-25', 'S40-25', 'S60-25', 'S20-50', 'S40-50', 'S60-50')

$I7$:... and sensor in ('S20-0', 'S40-0', 'S60-0', 'S20-25', 'S40-25', 'S60-25', 'S20-50', 'S40-50', 'S60-50', 'S20-75', 'S40-75', 'S60-75')

Table 3
Scaling up slices.

Intention	$I1$	$I2$	$I3$	$I4$	$I5$[a]	$I6$[a]	$I7$[a]
$\|C\|$	146	219	292	365	438	657	876
Time (s)	67.7	137.3	184.3	474.2	133.9	209.5	267.3
Highl. interest	0.91	0.91	0.91	0.91	0.91	0.99	0.99
♯ pred. values	7	10	14	18	21	32	43

[a] MultivariateTS is not computed for the sake of time.

Table 4
Scaling up slices: time breakdown by model (in seconds).

Model\$|C|$	146	219	292	365	438	657	876
regressionTree	0.2	0.2	0.2	0.2	0.2	0.3	0.3
randomForest	0.5	0.5	0.5	0.5	0.5	0.5	0.5
timeRegressionTree	0.4	0.7	1.0	1.1	1.3	2.0	2.6
timeRandomForest	1.0	1.5	1.9	2.4	2.9	4.3	5.6
univariateTS	46.7	68.2	89.7	108.3	128.9	202.5	258.3
multivariateTS	18.9	66.2	91.0	361.7	–	–	–

Adding sensors to the for clause increases the number of slices considered; note that all intentions must consider at least two sensors in order to compute multivariateTS (i.e., with a single slice/sensor we cannot have a multivariate time series). Since soilMoisture values were sampled in 2022, for every sensor we set 5% of the values to null in order to impute/forecast them.

Table 3 and Fig. 13 (left) summarize the results ($|C|$ is the cardinality of the cube, i.e., the number of facts, as determined by the for clause). The operator always returns good highlights (i.e., highlights with high interest), meaning that it can predict patterns recurrent in the training set. As to the time necessary to compute the intention, we start by observing that, in all cases, most of the time is taken to compute the models (i.e., by the **train and predict** phase, step 3 of the execution plan in Section 4.2) whereas the time to query the cube is negligible. For instance, in $I4$, getting a cube of 365 tuples takes 0.08 s out of 474.2 s to compute the intention. In turn, the time to compute the models is dominated by univariateTS and multivariateTS. From $I5$ to $I7$, we stopped computing the multivariateTS model since the time required is in the order of hours even for cubes with cardinalities of hundreds of cells; note that this is fully compliant with the goals of OLAP, whose returned datasets must be small enough to be manageable by humans for decision making [25].

Table 4 and Fig. 13 (right) report the time breakdown by model, showing that the computation time required by time-aware models scales linearly with the number of inspected measures (i.e., the number

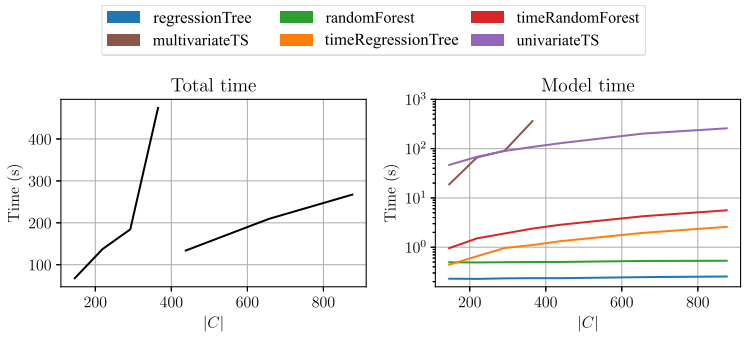


Fig. 13. Scaling up slices: total time (left) and breakdown by model (right).

Table 5
Scaling up the time span.

Intention	$I1$	$I2$	$I3$	$I4$	$I5$	$I6$	$I7$
$\|C\|$	1164	2604	4044	5484	6924	8364	12 684
Time (s)	297.5	522.4	735.0	939.3	1059.3	1525.4	1913.3
Highl. interest	1.0	0.9	0.9	0.9	0.7	0.9	0.8
♯ pred. values	58	130	202	274	346	418	634

Table 6
Scaling up the time span: time breakdown by model (in seconds).

Model\$|C|$	1164	2604	4044	5484	6924	8364	12 684
regressionTree	0.3	0.3	0.3	0.3	0.4	0.4	0.5
randomForest	0.5	0.6	0.7	0.7	0.8	0.9	1.1
timeReg.Tree	2.6	2.6	2.6	2.7	2.7	2.7	2.8
timeRand.For.	5.6	5.8	5.9	6.0	5.8	6.0	6.2
univariateTS	288.5	513.2	725.5	929.6	1049.6	1515.4	1902.8

of sensors selected in the for clause); since the y-axis is logarithmic, linear patterns follow logarithmic trends. This is because time-aware models pivot the base cube by the hour temporal attribute, and by construction they do not mix soil-moisture values of different sensors during the prediction. On the other hand, multivariateTS learns relationships among soil-moisture values of different sensors but its training time is way longer.

5.2.2. Scaling up the time span

To check how the performance scales when increasing the time span covered we run the following intentions, which predict the values of soilMoisture by hour and sensor in a given field for an increasing number of hours (all 12 sensors are considered):

with watering predict soilMoisture by hour, sensor for field = 'Field-1'...

$I1$:... and hour between '2022-07-0110 : 00 : 00' and '2022-07-0510 : 00 : 00'

$I2$:... and hour between '2022-07-0110 : 00 : 00' and "2022-07-1010 : 00 : 00'

$I3$:... and hour between '2022-07-0110 : 00 : 00' and "2022-07-1510 : 00 : 00'

$I4$:... and hour between '2022-07-0110 : 00 : 00' and "2022-07-2010 : 00 : 00'

$I5$:... and hour between '2022-07-0110 : 00 : 00' and "2022-07-2510 : 00 : 00'

$I6$:... and hour between '2022-07-0110 : 00 : 00' and "2022-07-3010 : 00 : 00'

$I7$:... and hour between '2022-07-0110 : 00 : 00' and "2022-08-1510 : 00 : 00'

Again, for every sensor we set 5% of the values to null in order to impute/forecast them. Besides, we let the operator compute all the models except multivariateTS since, as already mentioned, its computation time is too long above 1000 tuples.

Table 5 and Fig. 14 (left) summarize the results. The operator returns good highlights, with 0.66 being the minimum interest. The time necessary to compute the prediction is dominated by univariateTS (Table 6), the only model requiring more than 10 s. This means that, even after a few seconds, the predict operator can return a highlight, which can be incrementally improved as soon as new components are available. Even in this case, the time to query the cube is negligible (e.g., in $I7$, 0.4 s out of 1913.3 s to compute the intention).

We finally note that the slope for timeRegressionTree and timeRandomForest is less steep than for the other models. This is because $|C|$ is the cardinality of the whole cube, but time-aware models are computed on a cube pivoted by the temporal level; then, the increase of tuples *per component* (there is one component for each sensor) is actually lower than the overall increase of cube cardinality.

5.3. Summary

Overall, the experiments conducted reveal that the predict operator can return accurate predictions due to the complementarity of the adopted models. In the case of time-aware predictions (when time is not involved in the by clause of the intention), univariateTS and multivariateTS usually make better predictions than classical machine learning models adapted to time series (e.g., timeRegressionTree and timeRandomForest). However, univariateTS and (especially) multivariateTS are computationally heavier and require more time to make predictions. regressionTree and randomForest are required for time-agnostic predictions, and they also return accurate insights in the case of temporal queries; although regression trees are simpler than random forests, their adoption is worthwhile since they are more interpretable.

Since interactivity is a key feature of OLAP sessions, we believe that adopting an incremental approach where faster highlights with simpler models (such as timeRegressionTree and timeRandomForest) are returned first, to be later refined with more complex models, can be an added value for end users. In this regard, we show in Fig. 15 how the highlight interest increases with time, as model components are incrementally computed for intention $I2$ in Section 5.1.3. The first highlight (of the timeRegressionTree model type and with an interest of 0.92) is computed in 0.23 s. This is later refined by randomForest and timeRandomForest; the best highlight is computed in 26 s by univariateTS (multivariateTS is not shown since its interest is slightly lower than univariateTS). We also verified that involving additional related cubes in an intention can significantly improve the prediction quality.

Finally, we found that the predict operator is capable of retrieving patterns that are present in the data, e.g., the relationships between the adults and cumDegreeDays measures in the CAPTURE cube and the relationships between sensors (slices) in the WATERING cube.

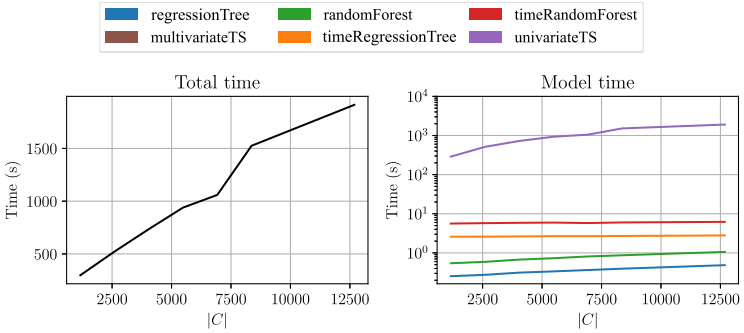


Fig. 14. Scaling up the time span: total time (left) and breakdown by model (right).

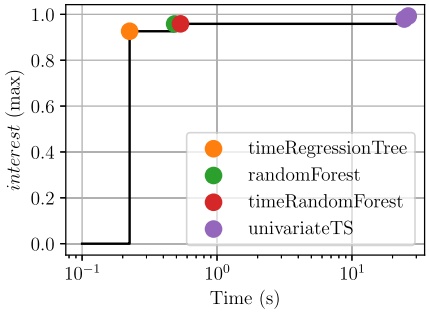


Fig. 15. Highlight interest as a function of time in incremental computation.

6. Related work

6.1. OLAP + analytics

The idea of coupling data and analytical models was born in the 90's with inductive databases, where data were coupled with patterns meant as generalizations of the data [26]. Later on, data-to-model unification was addressed in MauveDB [27], which provides a language for specifying model-based views of data using common statistical models. However, achieving a unified view of data and models was still seen as a research challenge in business intelligence a few years later [28]. More recently, Northstar [29] has been proposed as a system to support interactive data science by enabling users to switch between data exploration and model building.

The coupling of the OLAP paradigm and data mining to create an approach where concise patterns are extracted from multidimensional data for user's evaluation, was the goal of some approaches commonly labeled as OLAM [30]. In this context, k-means clustering is used in [31] to dynamically create semantically-rich aggregates of facts other than those statically provided by dimension hierarchies. Other operators that enrich data with knowledge extraction results are DIFF [32], which returns a set of tuples that most successfully describe the difference of values between two facts of a cube, and RELAX [33], which verifies whether a pattern observed at a certain level of detail is also present at a coarser level of detail. In [34] the authors reuse the OLAP paradigm to explore prediction cubes, i.e., cubes where each fact summarizes a predictive model trained on the data corresponding to that fact. Finally, the coupling of data and models is at the core of the IAM vision [9], on which this paper relies. The three basic pillars of IAM are (i) the redefinition of query as expressing the user's intention rather than explicitly declaring what data are to be retrieved, (ii) the extension of query results from plain data cubes to cubes enhanced with models and highlights, and (iii) the characterization of model components in terms of their interest to users.

6.2. Regression-based prediction

Regression analysis estimates the relationships between dependent and independent variables. It is used for prediction and forecasting, where its implementation substantially overlaps with machine learning techniques. Among these, we find a plethora of techniques ranging from decision-tree [13] and random-forest [14] regressors (the former is more interpretable, and the latter overcomes overfitting with boosting, bagging, and using multiple regression trees) to deep neural networks.

Specific techniques can be adopted when predictions are based on time series (a series of data points indexed in time order). The core of time series analysis is about discovering relationships and recurrent patterns in the data to predict future values based on historical observations, which provide the basis for decision-making processes. The application of time series has spanned various fields, such as agriculture and industry. The field of time series has witnessed many contributions over the years, some related to "classical" machine learning models [35], others related to the most recent advancements in deep learning, transformers, and generative models [36]. Among well-known approaches, we mention SARIMAX [15] and VARMAX [16]. SARIMAX is a technique for predicting univariate time series; the models and parts used for prediction are Seasonal, AutoRegression, dIfferencing, Moving Average, and eXogeneous variables (in addition to the endogenous variables). VARMAX is a technique for predicting multivariate time series; the models and parts used for prediction are Vector, AutoRegression, Moving Average, and eXogenous variables. Besides the type of the adopted technique, the main goal of these contributions is to make predictions (or anomaly detection) as accurate as possible, even by creating ensembles of existing models in an AutoML fashion [37]. This differs from the scope of our contribution. Our goal is not to deliver more accurate predictions than the state of the art but to create a formal framework that encapsulates models for time series analysis and multidimensional cubes, so that analysts can express high-level forecasting intentions that are automatically translated into execution plans involving (i) queries, (ii) machine learning models, (iii) interest computation, and (iv) component ranking and selection of the highlight. To the best of our knowledge, none of the contributions that have been recently surveyed in [35,36] falls in this scope, but still they can be leveraged as alternative models to be incorporated in the predict operator.

In *multilevel regression*, the presence of hierarchies of attributes is taken into account as well when making predictions (e.g., two products of the same type may have sales that are more similar than the sales of two products of different types). However, as shown in [38], this usually requires a preliminary feature engineering step to avoid a large increase of the feature space, so at to maintain good performances. For instance, it is necessary to indicate how (i.e., using which operator) detailed data can be aggregated and, to avoid computing all possible aggregations (whose number is exponential in the number of attributes), which group-by sets are actually meaningful. Since a requirement of our approach is to preserve the interactivity of OLAP sessions, we do

not adopt multilevel techniques so that our predictions are only based on the exact group-by set specified in the intention.

As to the models we adopt for our operator, we avoided techniques based on deep neural networks since they require a lot of time for their training; this is even worse in the case of analyses at different temporal scales. Indeed, having a long training before the operator can be used would conflict with the plug-and-play philosophy of our approach.

7. Conclusion

In this paper we made a significant step towards completing the IAM framework by introducing a syntax and a semantics for the predict operator, whose goal is to estimate the unknown values of a target measure based on the available knowledge. To make a prediction we make combined use of three types of regression models: regression trees, random forests, and time series. When a prediction intention is executed, first a set of data is selected from one or more related cubes and aggregated, then one or more models are computed with their components; finally, the most interesting component(s) are labeled as the highlight and coupled with the cube to form an enhanced cube.

The experiments we conducted with real datasets show that the predict operator is capable of returning good predictions thanks to the complementarity of the models and to the possibility of considering additional data stored in related cubes. The performances are compliant with the interactivity requirement of OLAP sessions; an exception is due to multivariate time series models, whose computation may require hours even for relatively small cubes.

Our future work will follow several directions:

- Devise a visualization metaphor for enhanced cubes that, while being compliant with the ones used for the other IAM operators [10–12], can consistently merge components belonging to different models to give the user a comprehensive view of the prediction.
- Expand the operator syntax by (i) allowing more than one target measure and more than one temporal index in the intention, and (ii) enabling, besides forecasting and imputation, also *allocation*, where a value of the target measure at some given aggregation level is allotted at a finer aggregation level. For instance, in our working example, this could be used to predict captures at the trap level based on captures at the province level.
- Improve the precision of the prediction by computing it using, instead of the group-by set specified in the by clause, a finer group-by set.
- Adopt an incremental approach to deliver the highlight, i.e., initially return the highlight obtained by computing the faster models, then progressively refine it in case more interesting highlights are produced by the slower models.
- The accuracy of a prediction depends on the ratio of the cardinality of the training set and the number of inspected measures and slices involved in the intention. To avoid wasting time in computing inaccurate models, the execution plan of predict could be expanded by including a check that, when this ratio is below a given threshold, suggests to change the intention by reducing the number of inspected measures and/or increasing the span of the for clause.

CRediT authorship contribution statement

Matteo Francia: Writing – original draft, Software, Methodology, Investigation, Data curation, Conceptualization. **Stefano Rizzi:** Writing – original draft, Methodology, Formal analysis, Conceptualization. **Matteo Golfarelli:** Writing – review & editing, Supervision, Methodology. **Patrick Marcel:** Writing – review & editing, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] S. Idreos, O. Papaemmanouil, S. Chaudhuri, Overview of data exploration techniques, in: Proc. SIGMOD, Melbourne, Australia, 2015, pp. 277–281.
[2] B. Tang, S. Han, M.L. Yiu, R. Ding, D. Zhang, Extracting top-k insights from multi-dimensional data, in: Proc. SIGMOD, Chicago, IL, USA, 2017, pp. 1509–1524.
[3] E. Zgraggen, Z. Zhao, R.C. Zeleznik, T. Kraska, Investigating the effect of the multiple comparisons problem in visual analysis, in: Proc. CHI, Montreal, QC, Canada, 2018, p. 479.
[4] L. Geng, H.J. Hamilton, Interestingness measures for data mining: A survey, ACM Comput. Surv. 38 (3) (2006) 1–32.
[5] P. Marcel, V. Peralta, P. Vassiliadis, A framework for learning cell interestingness from cube explorations, in: Proc. ADBIS, 2019, pp. 425–440.
[6] A. Ziegler, E. Kalliamvakou, X.A. Li, A. Rice, D. Rifkin, S. Simister, G. Sittampalam, E. Aftandilian, Measuring GitHub Copilot's impact on productivity, Commun. ACM 67 (3) (2024) 54–63.
[7] M. Francia, E. Gallinucci, M. Golfarelli, S. Rizzi, VOOL: A modular insight-based framework for vocalizing OLAP sessions, Inf. Syst. 129 (2025) 102496.
[8] P.J. Denning, Can generative AI bots be trusted? Commun. ACM 66 (6) (2023) 24–27.
[9] P. Vassiliadis, P. Marcel, S. Rizzi, Beyond roll-up's and drill-down's: An intentional analytics model to reinvent OLAP, Inf. Syst. 85 (2019) 68–91.
[10] M. Francia, P. Marcel, V. Peralta, S. Rizzi, Enhancing cubes with models to describe multidimensional data, Inf. Syst. Front. 24 (1) (2022) 31–48.
[11] M. Francia, M. Golfarelli, P. Marcel, S. Rizzi, P. Vassiliadis, Suggesting assess queries for interactive analysis of multidimensional data, IEEE Trans. Knowl. Data Eng. 35 (6) (2023) 6421–6434.
[12] M. Francia, S. Rizzi, P. Marcel, Explaining cube measures through intentional analytics, Inf. Syst. 121 (2024) 102338.
[13] W.-Y. Loh, Classification and regression trees, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 1 (1) (2011) 14–23.
[14] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[15] C. Chatfield, H. Xing, The Analysis of Time Series: An Introduction with R, CRC Press, 2019.
[16] R. Östermark, H. Saxén, VARMAX-modelling of blast furnace process variables, European J. Oper. Res. 90 (1) (1996) 85–101.
[17] M. Golfarelli, S. Rizzi, Data Warehouse Design: Modern Principles and Methodologies, McGraw-Hill, 2009.
[18] M. Terrovitis, P. Vassiliadis, S. Skiadopoulos, E. Bertino, B. Catania, A. Maddalena, S. Rizzi, Modeling and language support for the management of pattern-bases, Data Knowl. Eng. 62 (2) (2007) 368–397.
[19] R.G.D. Steel, J.H. Torrie, Principles and Procedures of Statistics, with Special Reference To the Biological Sciences, McGraw-Hill, New York, 1960.
[20] J. Korstanje, Advanced Forecasting with Python, Springer, 2021.
[21] D. Khider, F. Zhu, Y. Gil, autoTS: Automated machine learning for time series analysis, in: Proc. AGU Fall Meeting, San Francisco, CA, 2019, pp. PP43D–1637.
[22] M. Francia, E. Gallinucci, M. Golfarelli, COOL: A framework for conversational OLAP, Inf. Syst. 104 (2022) 101752.
[23] S. Jain, D. Moritz, D. Halperin, B. Howe, E. Lazowska, SQLShare: Results from a multi-year SQL-as-a-Service experiment, in: Proc. SIGMOD, San Francisco, CA, USA, 2016, pp. 281–293.
[24] M. Francia, J. Giovanelli, M. Golfarelli, Multi-sensor profiling for precision soil-moisture monitoring, Comput. Electron. Agric. 197 (2022) 106924.
[25] M. Francia, M. Golfarelli, S. Rizzi, A-BI^{+}: A framework for augmented business intelligence, Inf. Syst. 92 (2020) 101520.
[26] L.D. Raedt, A perspective on inductive databases, SIGKDD Explor. 4 (2) (2002) 69–77.
[27] A. Deshpande, S. Madden, MauveDB: supporting model-based user views in database systems, in: Proc. SIGMOD, 2006, pp. 73–84.
[28] T.B. Pedersen, Warehousing the world: A vision for data warehouse research, in: S. Kozielski, R. Wrembel (Eds.), New Trends in Data Warehousing and Data Analysis, Vol. 3, Springer, 2009, pp. 1–17.
[29] T. Kraska, Northstar: An interactive data science system, PVLDB 11 (12) (2018) 2150–2164.
[30] J. Han, OLAP mining: Integration of OLAP with data mining, in: Proc. Working Conf. on Database Semantics, 1997, pp. 3–20.

[31] F. Bentayeb, C. Favre, RoK: Roll-up with the k-means clustering method for recommending OLAP queries, in: Proc. DEXA, 2009, pp. 501–515.
[32] S. Sarawagi, Explaining differences in multidimensional aggregates, in: Proc. VLDB, Edinburgh, Scotland, 1999, pp. 42–53.
[33] G. Sathe, S. Sarawagi, Intelligent rollups in multidimensional OLAP data, in: Proc. VLDB, Rome, Italy, 2001, pp. 531–540.
[34] B. Chen, L. Chen, Y. Lin, R. Ramakrishnan, Prediction cubes, in: Proc. VLDB, Trondheim, Norway, 2005, pp. 982–993.
[35] Z. Liu, Z. Zhu, J. Gao, C. Xu, Forecast methods for time series data: A survey, IEEE Access 9 (2021) 91896–91912.
[36] Y. Liang, et al., Foundation models for time series analysis: A tutorial and survey, in: Proc. SIGKDD, Barcelona, Spain, 2024, pp. 6555–6565.
[37] G. Westergaard, et al., Time series forecasting utilizing automated machine learning (AutoML): A comparative analysis study on diverse datasets, Inf. 15 (1) (2024) 39.
[38] P.C. Austin, J. Merlo, Intermediate and advanced topics in multilevel logistic regression analysis, Stat. Med. 36 (20) (2017) 3257–3277.