Automatic Machine Learning-based OLAP Measure Detection for Tabular Data

Yuzhao Yang, Fatma Abdelhedi, Jérôme Darmont, Franck Ravat, Olivier Teste

Automatic Machine Learning-based OLAP Measure Detection for Tabular Data

Yuzhao Yang[1], Fatma Abdelhédi[3], Jérôme Darmont[2], Franck Ravat[1], and Olivier Teste[1]

[1] IRIT-CNRS (UMR 5505), Université de Toulouse, France
{Yuzhao.Yang, Franck.Ravat, Olivier.Teste}@irit.fr
[2] Univ Lyon, Univ Lyon 2, UR ERIC, France
jerome.darmont@univ-lyon2.fr
[3] CBI[2] – TRIMANE, Paris, France
Fatma.Abdelhedi@trimane.fr

Abstract. zhang2011handling combining naive bayes and em, li2015trip combining dependencies and web information, zhang2019learning combining KNN and Regression, song2020imputing combining KNN and likelihood maximization, chhabra2018missing combining k-means and association rules, aydilek2013hybrid combining fuzzy c-means, support vector regression and genetic algorithm, etc. There are also hybrid approaches which use techniques of the same category like latifi2012evaluation combining knn and random forest and wang2017cosset combining crowdsourcing and knowledge base, etc.
Nowadays, it is difficult for companies and organisations without Business Intelligence (BI) experts to carry out data analyses. Existing automatic data warehouse design methods cannot treat with tabular data commonly defined without schema. Dimensions and hierarchies can still be deduced by detecting functional dependencies, but the detection of measures remains a challenge. To solve this issue, we propose a machine learning-based method to detect measures by defining three categories of features for numerical columns. The method is tested on real-world datasets and with various machine learning algorithms, concluding that random forest performs best for measure detection.

Keywords: Data warehouses, OLAP, Measure detection, Tabular data

1 Introduction

Business Intelligence (BI) plays an important role in numerous companies and organizations to efficiently support decision making processes. In classical BI architectures, data from heterogeneous sources are integrated into a Data Warehouse (DW) usually modeled in a multidimensional way, allowing decision makers to analyze data by On-Line Analytical Processing (OLAP) [11]. A multidimensional DW organizes data according to analysis subjects (facts) associated with analysis axes (dimensions). Dimension attributes may be ordered according to their granularity (hierarchies) and each fact is composed of indicators (measures).

With the development of information systems and the availability of numerous open datasets, various data become much more accessible to enterprises, organizations and even individuals, who have data analysis needs to help them take decisions [1]. However, the design of a DW is typically carried out manually, requires expert knowledge and BI experience [26], may be time-consuming and costly. Thence, automating the DW design process is desirable to allow businesses and organizations take advantage of BI.

There are different automatic or semi-automatic approaches for the design of multidimensional DW schemas [25]. However most of these methods focus on data sources with explicit schema: relational data with Entity-Relationship (ER) schema, XML data with Document Type Definitions (DTDs), etc. Nevertheless, tabular data such as spreadsheet data and Comma Separated Value (CSV) files are very common in enterprises, and even more in the open data world. They may also be DW data sources, but whose schema is not available. Building dimensions and hierarchies for tabular data can be done by detecting functional dependencies [32]. In the existing methods for other sources, measures are defined manually by users or are detected with respect to data types (numerical values) and cardinalities, which is impractical for tabular data without schema. Yet, measures remain central elements in multidimensional models, as they are the indicators that assess the analyzed activities. **Therefore, measure detection for automatic DW design from tabular data is an important task**. To the best of our knowledge, there is no specific approach addressing this challenge.

Tabular data may bear quite simple or very complex structures [2]. Simple structures consist of one header row followed by rows containing data values. Headers label the data rows below, while data rows contain tuples akin to relational database tuples. Most CSV files bear a simple structure, while spreadsheet files and HTML tables can be more complex, e.g., cross tables [17]. Such tables contain two or several dimensions, and may also contain several dimension levels. Moreover, there also exists other complex structures such as concise tables, nested tables, multivalued tables and split tables [17].

The data region can be extracted from a cross tables by some algorithms [5,6,16,31] where measures are located. The other types of complex structures can be converted into simple structures [6]. However, for simple-structured tabular data, DW elements cannot be directly extracted either without a schema or metadata, as the data do not bear a particular layout. Measures are usually numerical data, but numerical columns are not necessarily measures, since there also exists descriptive numerical attributes. Thus, we intend to find the numerical columns that conform to the characteristics of measures. We hypothesize that there are differences in terms of features between numerical data that are potential measures and those that are not. **Therefore, in this paper, we define specific features for numerical columns and propose a machine learning-based method to automatically detect measures.**

The remainder of this paper is organized as follows. In Section 2, we review the related works about measure selection for automatic DW design. In Section 3, we detail and discuss the measure detection process and the features we

propose. In Section 4, we present and interpret our experimental results. Finally, in Section 5, we conclude this paper and hint at future research.

2 Related Works

There are various methods dedicated to automatic or semi-automatic DW schema design, with different measure selection approaches. Since the selected measures should correspond to business requirements, in many methods [8,9,14,27], they are assigned directly by the users.

In a semi-automatic method to model DW from E/R diagrams [13], the fact table is selected by calculating the Connection Topology Value (CTV) of each entity, which is a composite function of the topology value of direct and indirect one-to-many relationships. Measures are still chosen manually by the user, but the scope of the choice is reduced. Moreover, another approach fully automates DW design from an E/R schema [22]. In contrast, some approaches aim at discovering measures, including 1) selecting many-to-many relationships containing numeric and additive non-key facts [15]; 2) analyzing business queries for data items indicating business performance [4]; 3) basing on most frequently updated entities [10]; or 4) selecting the numerical data that can be aggregated [30]. All these approaches work in the context of automatic DW design based on E/R schemas. However, the constraints and relationships mentioned in such methods cannot be directly applied on tabular data.

Another trend is using knowledge-based methods for automatic DW design [28]. Key information on measures and dimensions are extracted through a Natural Language Processing (NLP) model based on sentences from the business requirements. Candidate measures are all numerical column and are then validated by some constraints in a predefined domain ontology and by checking whether they can be aggregated. However, the method needs business requirements to train the NLP model. Defining the domain ontology is also difficult.

In summary, there is no specific method to automatically detect measures from tabular data in the absence of schema and explicit business requirements. Thus, we propose a machine learning-based method for measure detection from tabular data.

3 Measure Detection

3.1 Overview

Figure 1 shows an overview of our measure detection process for tabular data. If tabular data bear a complex structure, we use table structure detection algorithms [5,6,16,31] to verify whether data lie in a cross table. If so, measures are extracted from the data region, save aggregated values are excluded. Otherwise, data are converted by the algorithm proposed in [6] into a simple structure that is formally defined in Definition 2.

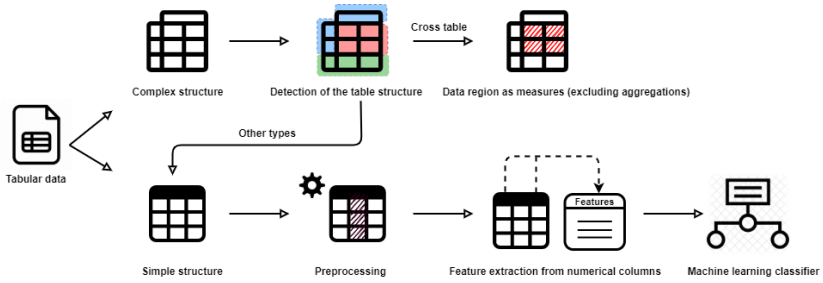

Fig. 1: Measure detection for tabular data

Definition 1. *Measures are numerical and quantitative attributes of the analysis subject evaluating the activities of an organisation and that can be aggregated with respect to dimensions. They can be additive, semi-additive or non-additive [12].*

Definition 2. *A tabular dataset of simple structure* $\boldsymbol{TS}$ *is defined as* $\{\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{A}, \boldsymbol{V}\}$*, where:*

- $\boldsymbol{C} = \{C_1, C_2, ...C_{n_c}\}$ *is a set of columns, where* n_c *is the number of columns in* $\boldsymbol{TS}$*. For a given column* $C_i \in \boldsymbol{C}$*, index* i *corresponds to the column's position in* $\boldsymbol{TS}$*. The number of non-null values in column* C_i *is denoted as* $n_t(C_i)$*. The number of non-null distinct values is denoted as* $n_u(C_i)$*;*
- $\boldsymbol{R} = \{R_1, R_2, ..., R_{n_r}\}$ *is a set of rows (excluding the first, header row), where* n_r *is the number of non-header rows in* $\boldsymbol{TS}$*. For a given row* $R_j \in \boldsymbol{R}$*,* j *represents the index of the row corresponding to its position in* $\boldsymbol{TS}$*;*
- $\boldsymbol{A} = \{A_{C_1}, A_{C_2}, ..., A_{C_{n_c}}\}$ *is a set of attribute headers. For a given attribute header* $A_{C_i} \in \boldsymbol{A}$*,* C_i *represents the column labeled by* A_{C_i}*;*
- $\boldsymbol{V}$ *is a matrix of cell values whose dimension is* $n_r \times n_c$*. For a given cell value* $V_{R_j,C_i} \in \boldsymbol{V}$*,* R_j *and* C_i *are the row and the column where the cell is located, respectively.*

In the following sections, we focus on measure detection for tabular data of simple structure. Since measures are numerical, we regard all numerical columns as candidates. Yet, preprocessing the dataset is necessary for the selection of numerical columns. Then, to distinguish between measure and non-measure numerical columns we extract features from numerical columns and use machine learning classifiers to estimate whether they are measures.

3.2 Preprocessing

As candidate measures are numerical columns, we must first identify numerical columns. If all values of a column are numerical, we easily identify numerical columns. However, there are sometimes columns containing numerical values with their unit, or columns containing both numerical and textual values used

for replacing empty cells. Such mixed values must lead to numerical columns and require preprocessing.

Columns containing values with a unit are identified by verifying whether each cell obear the same structure, e.g., "text + number" or "number + text". We also verify whether the text of each column is the same or if it is categorical by using the algorithm proposed by [3]. Then, we extract numerical values via regular expressions and tag the column as numerical. Eventually, numerical columns containing empty values replaced by some text, e.g., "n/a", "null" or "unknown", are treated as numerical, with textual values being removed.

3.3 Feature Extraction

After the preprocessing phase, we extract the numerical columns' features. When defining features, we analyze both general information and some statistical characteristics of numerical columns. Since tabular data of simple structure are usually relational and may exhibit specific column positional habits, we also consider column inter-relationships. Features are thus subdivided into three categories: general features, statistical features and inter-column features. For a given numerical column C_i, we define the following features.

General Features These features reflect basic information on numerical columns. Such general features may help check whether a numerical column is likely to be quantitative and help evaluate business activities. General features follow.

- **Data type**: $type = \begin{cases} 1 \text{ if } type(C_i) = integer \\ 0 \text{ if } type(C_i) = float \end{cases}$, where $type(C_i)$ is C_i's data type.

Intuitively, float data are more likely to be quantitative and to allow evaluating activities. For example, temperature, salary and sales amount are float data can be considered as measures in most cases.

- **Positive/Negative/Zero value ratio**: $rpos = \dfrac{n_{pos}(C_i)}{n_t(C_i)}$, $rneg = \dfrac{n_{neg}(C_i)}{n_t(C_i)}$, $rzero = \dfrac{n_{zero}(C_i)}{n_t(C_i)}$, where $n_{pos}(C_i)$, $n_{neg}(C_i)$ and $n_{zero}(C_i)$ are the number of positive, negative and zero values in C_i, respectively, and $n_t(C_i)$ is the number of non-null values in C_i.

These features may help identifying both qualitative and quantitative columns. Qualitative data values, e.g., ID or zip code, are rarely negative or equal to zero. Thus, when there are many zero and negative values in a column, it is more likely to be a measure.

- **Unique value ratio**: $runique = \dfrac{n_u(C_i)}{n_t(C_i)}$.

The unique value ratio can reveal some typological information about a column. For example, in a descriptive dataset, IDs are always unique, so the unique value ratio is always 1. In a dataset containing fact table data, keys and descriptive data may be repetitive, but equal measures should be quite scarce.

- **Same digital number**:
  $sdn = \begin{cases} 1 \text{ if } \forall i \in [1, n_t(C_i) - 1], nd_{R_j,C_i} = nd_{R_{j+1},C_i} \wedge type(C_i) = integer \\ 0 \text{ if } (\exists i \in [1, n_t(C_i) - 1], nd_{R_j,C_i} \neq nd_{R_{j+1},C_i} \wedge type(C_i) = integer) \\ \quad \vee (type(C_i) = float) \end{cases}$
  where nd_{R_j,C_i} is the number of digits in cell value V_{R_j,C_i}, which is calculated as $nd_{R_j,C_i} = floor(log_{10}^{V_{R_j,C_i}}) + 1$.

This feature tells whether all the values of an integer column have the same number of digits. If it is the case, the column is likely to be a nominal number [3] representing the name or identifier of an element that cannot be a measure. For example, the French social security number always contains 15 digits.

Statistical Features Since candidate columns are numerical, statistical features must be considered. They can indeed reflect the distribution of column values. Statistical features follow.

- **Average/Minimum/Maximum/Median/Upper quartile/Lower quartile values**: $avg = avg(C_i)$, $min = min(C_i)$, $max = max(C_i)$, $median = median(C_i)$, $upquar = upquar(C_i)$ and $lowquar = lowquar(C_i)$ represent the average, minimum, maximum, median, upper quartile and lower quartile of C_i, respectively.

We consider these basic statistical metrics as features. In some specific columns, their values always vary in a certain range. Using these features can thus be helpful for capturing such statistical behaviours.

- **Coefficient of variation**:
  $coevar = \begin{cases} standdev(C_i) \text{ if } avg(C_i) = 0 \\ \dfrac{standdev(C_i)}{avg(C_i)} \text{ if } avg(C_i) \neq 0 \end{cases}$
  where $standdev(C_i)$ is C_i's standard deviation.

The standard deviation can depict the amount of dispersion of a column values. Measures or descriptive attributes may have different degrees of dispersion, but by using the coefficient of variation, which is the ratio of the standard deviation by the average, we achieve a standardized degree of dispersion. For example, given two attributes "price of phone" and "temperature of city", the average price is much higher than that of temperature. A price variation of 10 is relatively much lower than that of tempera. Since the coefficient of variation is a ratio, when the average is equal to 0, it does not exist. Here, we define that when the average is 0, the feature is equal to the standard deviation of the column.

– **Range ratio**: $rrange = \dfrac{max - min}{n_u(C_i)}$.

The range ratio calculates the range of values with respect to the number of distinct values. It is useful to identify some ordinal data, even if they occur repetitively. For example, if we have student numbers ranging from 1000 to 2000 in a tabular dataset, but also courses and grades, a student number may occur many times while the range ratio is always 1 no matter the number of occurrences.

Inter-Column Features Measures are aggregatable and are normally accompanied with attributes by which they are aggregated, as per the "group by" SQL clause. Typically, attributes linked to aggregations are located before measures in the source file. Therefore, we consider inter-column features that take inter-column relationships into account in the whole dataset.

– **Location ratio**: $rloc = \dfrac{i-1}{n_c - 1}$.

In many tables, the identifier and some other basic information usually lie at the beginning positions, while measures are usually in the latter positions. Thus, we also take column location into account. However, different datasets have different number of columns, so we must normalize the location feature as a ratio ranging between 0 and 1.

– **Numerical column ratio**: $rnum = \dfrac{n_{num}}{n_c}$, where n_{num} is the number of numerical columns in the whole dataset.

The ratio of numerical column number by total column number is a table feature. While there are tabular data that only contain descriptive information, others include numerical columns that may be measures.

– **Multiple functional dependencies**:
$severalfds = \begin{cases} 1 \text{ if } \exists fd \in fdset, (fd.rhs = A_{C_i}) \wedge (size(fd.lhs) > 1) \\ 0 \text{ else} \end{cases}$
where $fdset$ is the functional dependency set of the dataset, $fd.rhs$ is the right hand side attribute of functional dependency fd and $size(fd.lhs)$ is the number of attributes in the left hand side of fd.

In existing methods that exploit data sources with schemas, many-to-many relationships are usually employed for measure detection. In a DW, we usually analyze a fact with respect to different dimensions and measure values depend on dimensions' primary keys. Thus, we consider whether there is a functional dependency with A_{C_i} depending on several attributes as a feature.

– **Numerical neighbor**:

$$numn = \begin{cases} 1 & \text{if } (i = 1 \wedge type(C_{i+1}) \in num) \vee (i = n_c \wedge type(C_{i-1}) \in num) \\ & \quad \vee (i \neq 1 \wedge i \neq n_c \wedge type(C_{i+1}) \in num \wedge type(C_{i-1}) \in num) \\ 0.5 & \text{if } (i \neq 1 \wedge i \neq n_c \wedge type(C_{i+1}) \in num \wedge type(C_{i-1}) \notin num) \\ & \quad \vee (i \neq 1 \wedge i \neq n_c \wedge type(C_{i+1}) \notin num \wedge type(C_{i-1}) \in num) \\ 0 & \text{else} \end{cases}$$

where $num = \{integer, float\}$.

In a tabular dataset, the columns describing similar information are often clustered together. Measures are also likely to be located close together, meaning that there are numerical columns in neighboring positions. Thus, we define this feature to see if neighbors of a column are also numerical. If so, the column is likely to be a measure.

4 Experimental Validation and Discussion

4.1 Experimental Conditions

Our experiments are conducted on an Intel(R) Core(TM) i5-10210U 1.60 GHz CPU with a 16 GB RAM. We use 9 datasets in our experiments, from the governmental open data sites of France (**FR**), Canada (**CA**), UK (**UK**) and US (**US**), the French Development Agency (**AFD**), the New Zealand's official data agency (**NZ**), the American Center for Disease Control and Prevention (**CDC**), the World Bank (**WB**) and Kaggle (**KG**). Each dataset contains numerous tables with numerical columns on which features are extracted to feed the algorithms. Moreover, they are classified into five domains (Table 3) including Economy (**ECO**), Health (**HLT**), Government (**GOV**), Environment (**ENV**) and Society (**SOC**). Complete information about these datasets are provided in Appendix.

We apply the following widely used Machine Learning (ML) classification algorithms [29] (available in Python 3.7): 1) an SVM classifier with an RBF kernel (**SVM**), 2) a decision tree classifier based on the CART algorithm (**DT**), 3) a random forest classifier (**RF**) and 4) a k-nearest neighbors classifier (**KNN**). Deep learning models are not employed because they are more suitable for interpreting images, sounds and texts [18], while we analyze numerical columns.

We define the ground truth by analyzing each dataset context according to its website's description, header semantics and metadata. We also uphold the criteria from Definition 1. Thence, for each dataset, we compute all our proposed features (Section 3.3) for each numerical column, and label them to build training and test sets. Empty values in columns are ignored and not counted.

4.2 Baseline Methods

Numerical Typology-Based Method (TP) In a previous work, we proposed to select measures with respect to the type of numerical attributes [32]. Numerical data may be classified into nominal data, ordinal data, intervals and ratios [3]. Algorithms can detect the different numerical types [3]. We identified the columns of interval and ratio types as DW measures.

Functional Dependency-Based Method (FDB) As we already mentioned, in existing methods aimed at data with schemas, measures are selected in tables exhibiting many-to-many relationships; in other words, columns that are functionally dependent on dimension primary keys. With this idea in mind, we detect functional dependencies (FDs) in tabular data and select as measures the numerical columns that are functionally determined by several, other attributes. The FD detection algorithm that we use is HyFD [21]. HyFD indeed achieves the best performance against the seven most cited and important algorithms that are tested in [19].

We employ the Metanome Web-based toolbox [20], which is developed by the HyFD designers, to implement HyFD. Moreover, we use the Python library selenium[4] to feed input files in Metanom and get the FDs automatically. The extracted FDs are also used for generating the values of feature **severalfds**.

4.3 Experimental Results

Algorithm Effectiveness We run the two baseline methods from Section 4.2 and train models with our proposed features by four ML algorithms (Section 4.1) on all datasets (Section 4.1). The ML algorithms are run by pycaret [5] AutoML Python library where the hyperparameters are tuned automatically. For the model generality and feauture importance experiments, we run ML algorithms from the sklearn[6] Python library.

We use three performance metrics: Recall (**R**), Precision (**P**) and F-Measure (**F**), as follows. Let N_{mm} and N_{mn} be the number of measures predicted as measures and non-measures, respectively; and N_{nm} and N_{nn} the number of non-measure predicted as non-measures and measures, respectively.

Then, $\mathbf{R} = \frac{N_{mm}}{N_{mm} + N_{mn}}$, $\mathbf{P} = \frac{N_{mm}}{N_{mm} + N_{nm}}$ and $\mathbf{F} = \frac{2 \times Precision \times Recall}{Precision + Recall}$.

Table 1 shows the resulting values of **R**, **P** and **F** where the results of ML algorithms are obtained through a 10-fold cross validation by merging all datasets and randomly split them into 10 folds. The distribution of the cross validation results is depicted in Figure 2.

We observe that **RF** exhibits the best F-measure (94.82%) and the result is not more dispersed than that of the other algorithms. Thus, **RF** shows the best performance on the measure detection problem. We also observe that **TP** and **FDB** do not have a good effectiveness when predicting measures, but **FDB** performs better than **TP**. **TP**'s bad performance is due to: 1) interval and ratio numerical columns are not all measures, e.g., longitude and latitude; 2) numerical typology detection algorithm are not flexible enough to cope with real-world data, because they are based on fixed rules. Regarding **FDB**, a numerical column that is functionally determined by several other columns may not always be a measure, either. For example, let us consider a table describing sale facts with

[4] `https://selenium-python.readthedocs.io`
[5] `https://pycaret.org/`
[6] `https://scikit-learn.org`

Table 1: Global results

	TP	FDB	RF	SVM	DT	KNN
R(%)	80.05	75.43	**96.64**	94.77	94.08	90.16
P(%)	73.57	77.50	**90.89**	78.44	88.44	87.61
F(%)	76.67	76.45	**93.65**	85.76	91.12	88.78

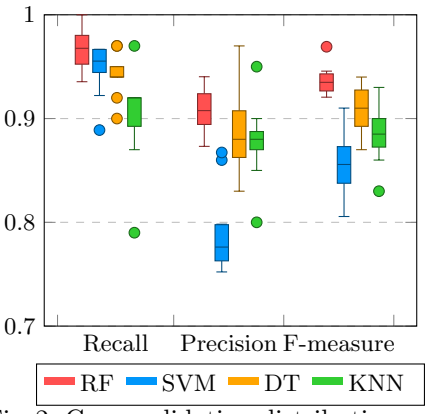

Fig. 2: Cross validation distribution

respect to customers and products, where sales' amount is indeed a measure. The customer ID is the customer dimension's primary key, but the customer's name and email may uniquely identify a customer, and thus may functionally determine the age of the customer, a numerical column that is not a measure.

Our ML-based measure detection method takes different types of features (Section 3.3) into account and can thus better handle the above exceptions and get better results.

Feature Category Effectiveness To verify the effectiveness of each feature category we propose, we test different combinations of feature categories with our **RF**-based method. We first test single feature categories, combinations of two categories and then we compare the effectiveness of all categories. The result is shown in Table 2, where **GE** represents GEneral features, **ST** represents STatistical features and **IC** represents Inter-Column features. **ST** exhibits the best individual contribution. Yet, we can clearly see that combining feature categories achieves better performance in terms of recall, precision and F-measure, than using single feature categories. Ultimately, combining all feature categories yields the best performance. The results of applying other ML algorithms can be found in our github.

Table 2: Performance of feature categories and combinations with **RF**

	GE	ST	IC	GE+ST	GE+IC	ST+IC	ALL
R(%)	88.10	94.27	92.68	95.30	93.67	91.93	**96.64**
P(%)	83.59	86.28	80.91	88.21	86.13	91.14	**90.89**
F(%)	85.69	90.01	86.37	91.57	89.67	91.50	**93.65**

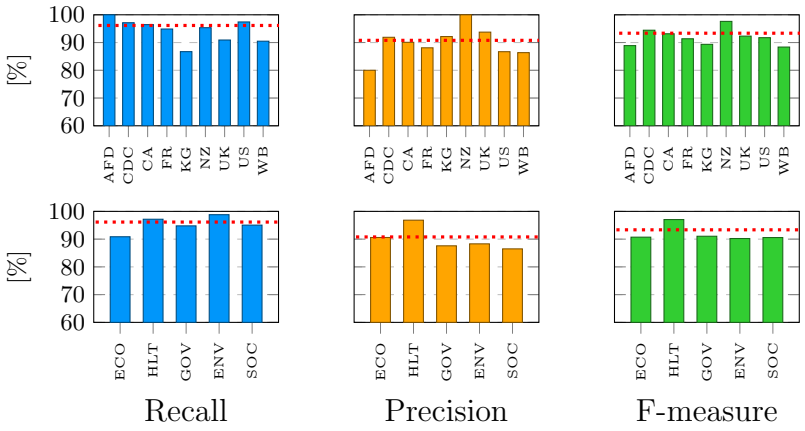

Fig. 3: Performance with respect to source and domain

Model Generality To verify that the trained model achieved with our **RF**-based method is generic, we train data by excluding the datasets of one sources and test on them. We also carry out the same test by domain, i.e., economy (**ECO**), health (**HLT**), government (**GOV**), environment (**ENV**) and society (**SOC**). The results are shown in Figure 3, where the charts above and below depict the results by source and domain, respectively. By comparing with former results, the difference of F-measure ranges from -5.02% to 4.23% for the test with respect to the source and from -3.17% to 3.36% for the test with respect to the domain. The trained model with the defined features is thus generic regardless of the source and the domain of data. The results of applying other ML algorithms can be found in our github[7].

Feature Importance To analyze our different features, we compute the permutation importance (decrease in prediction accuracy when a feature is permuted [7]) of each feature for all ML algorithms. Figure 4 shows that the importance of a feature varies with respect to the algorithm. For example, with **SVM** and **KNN**, some statistical features are more important than others, while with **RF** and **DT**, the features bearing the highest importance values are more equally distributed in each feature category. There are also features that bears negative importance values with algorithms, but not every time, while they always have positive importance values with other algorithms. There is no feature that always bears zero or negative importance values with one given algorithm, which means that all our features have a contribution to the ML classifiers. With **RF**, which bears the best performance, the most important feature is the location ratio. By checking the CSV files, we observe that most of the measures are situated at the last part of the file, while most of the columns in the front part are descriptive, which probably explains the importance of the location ratio.

[7] `https://github.com/Implementation111/measure-detection`

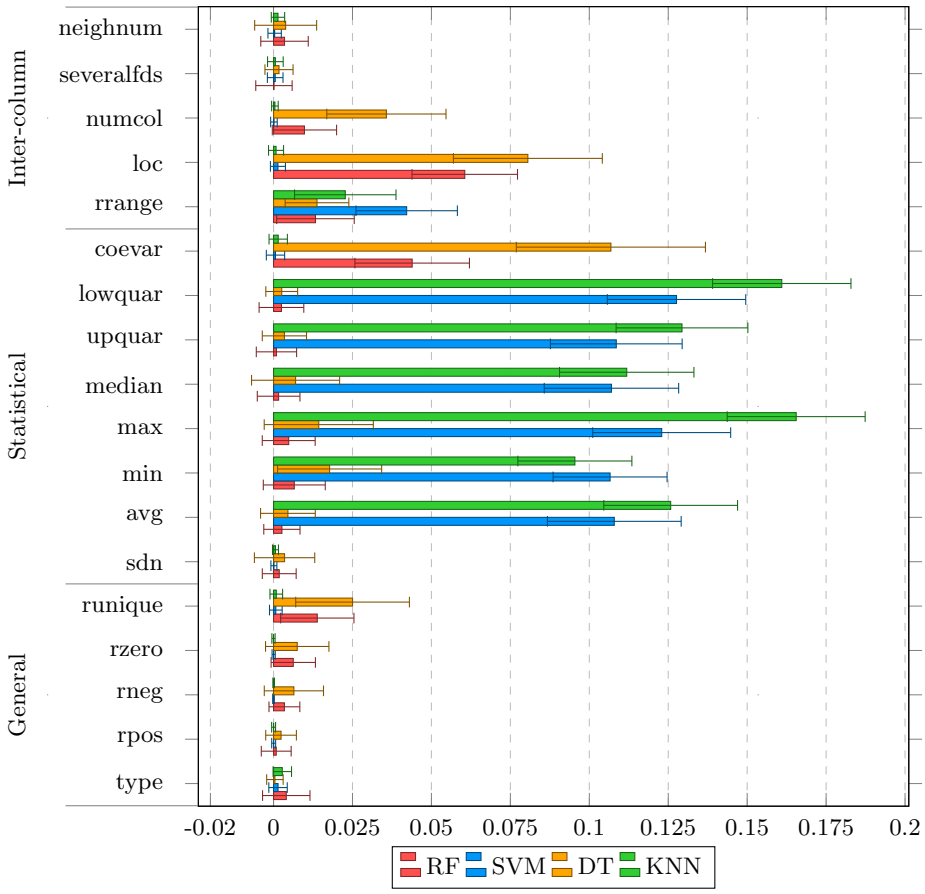

Fig. 4: Feature importance

5 Conclusion and Future Work

In this paper, we propose a machine learning-based method for detecting OLAP measures from tabular data. Our method is mainly dedicated to tabular data of simple structure, since the case of complex structures has been addressed in the literature. Some complex structures can also be converted to simple structures. To fuel machine learning algorithms, we define three categories of features for numerical columns in tabular data. We experiment with several real-world CSV datasets and test four machine learning algorithms, among which Random Forest performs the best. We also analyze how the features we build contribute to the results.

In the future, we aim at considering not only numerical measures, but also textual measures [23]. Moreover, we only use CSV data in our experiments. Com-

plementing them with other types of tabular data, including complex-structured data that are found in data lakes [24], could be relevant.

Acknowledgements

The research depicted in this paper is funded by the French National Research Agency (ANR), project ANR-19-CE23-0005 BI4people (Business Intelligence for the people).

References

1. Abelló, A., Darmont, J., Etcheverry, L., Golfarelli, M., Mazón, J.N., Naumann, F., Pedersen, T., Rizzi, S.B., Trujillo, J., Vassiliadis, P., Vossen., G.: Fusion cubes: Towards self-service business intelligence. International Journal of Data Warehousing and Mining **9**(12), 66–88 (2013)
2. Adelfio, M.D., Samet, H.: Schema extraction for tabular data on the web. VLDB Endowment **6**(6), 421–432 (2013)
3. Alobaid, A., Kacprzak, E., Corcho, O.: Typology-based semantic labeling of numeric tabular data. Semantic Web **1**, 1–5 (2019)
4. Ballard, C., Herreman, D., Schau, D., Bell, R., Kim, E., Valencic, A.: Data modeling techniques for data warehousing
5. Chen, Z., Cafarella, M.: Automatic web spreadsheet data extraction. In: 3rd International Workshop on Semantic Search Over the Web. pp. 1–8 (2013)
6. Du, L., Gao, F., Chen, X., Jia, R., Wang, J., Zhang, J., Han, S., Zhang, D.: Tabularnet: A neural network architecture for understanding semantic structures of tabular data. In: 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. p. 322–331 (2021)
7. Fisher, A., Rudin, C., Dominici, F.: All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. Journal of Machine Learning Research **20**(177), 1–81 (2019)
8. Giorgini, P., Rizzi, S., Garzetti, M.: Goal-oriented requirement analysis for data warehouse design. In: 8th ACM International Workshop on Data Warehousing and OLAP. p. 47–56 (2005)
9. Golfarelli, M., Rizzi, S., , Vrdoljak, B.: Data warehouse design from xml sources. In: 4th ACM international workshop on Data warehousing and OLAP. p. 40–47 (2001)
10. Golfarelli, M., Maio, D., Rizzi, S.: Conceptual design of data warehouses from e/r schemes. In: Proceedings of the thirty-first Hawaii international conference on system sciences. vol. 7, pp. 334–343 (1998)
11. Golfarelli, M., Rizzi, S.: Data Warehouse Design: Modern Principles and Methodologies. McGraw-Hill, Inc., 1 edn. (2009)
12. Horner, J., Song, I.Y., Chen, P.P.: An analysis of additivity in olap systems. In: 7th ACM International Workshop on Data Warehousing and OLAP. p. 83–91 (2004)
13. I.-Y.Song, Khare, R., Dai, B.: Samstar: A semi-automated lexical method for generating star schemas from an entity-relationship diagram. In: ACM 10th International Workshop on Data Warehousing and OLAP. pp. 9–16 (2007)
14. Jensen, M.R., Holmgren, T., Pedersen, T.B.: Discovering multidimensional structure in relational data. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) Data Warehousing and Knowledge Discovery. pp. 138–148 (2004)

15. Kimball, R.: A dimensional modeling manifesto. DBMS **10**(9), 58–70 (1997)
16. Koci, E., Thiele, M., Romero, O., Lehner, W.: A machine learning approach for layout inference in spreadsheets. In: International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management. p. 77–88 (2016)
17. Lautert, L.R., Scheidt, M.M., Dorneles, C.F.: Web table taxonomy and formalization. ACM SIGMOD Record **42**(3), 28–33 (2013)
18. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–444 (2015)
19. Papenbrock, T., Ehrlich, J., Marten, J., Neubert, T., Rudolph, J., Schönberg, M., Zwiener, J., Naumann, F.: Functional dependency discovery: An experimental evaluation of seven algorithms. In: VLDB Endowment. vol. 8, p. 1082–1093 (2015)
20. Papenbrock, T., Bergmann, T., Finke, M., Zwiener, J., Naumann, F.: Data profiling with metanome. VLDB Endowment **8**(12), 1860–1863 (2015)
21. Papenbrock, T., Naumann, F.: A hybrid approach to functional dependency discovery. In: International Conference on Management of Data. p. 821–833 (2016)
22. Phipps, C., Davis., K.C.: Automating data warehouse conceptual schema design and evaluation. In: 4th International Workshop on Design and Management of Data Warehouses. pp. 23–32 (2002)
23. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Top_keyword: An aggregation function for textual document olap. In: Data Warehousing and Knowledge Discovery. pp. 55–64 (2008)
24. Ravat, F., Zhao, Y.: Data lakes: Trends and perspectives. In: Database and Expert Systems Applications. pp. 304–313 (2019)
25. Romero, O., Abelló, A.: A survey of multidimensional modeling methodologies. International Journal of Data Warehousing and Mining **5**(2), 1–23 (2009)
26. Romero, O., Abelló, A.: A framework for multidimensional design of data warehouses from ontologies. Data & Knowledge Engineering **69**(11), 1138–1157 (2010)
27. Sandro Bimonte, Lucile Sautot, L.J., Faivre, B.: Multidimensional model design using data mining: A rapid prototyping methodology. International Journal of Data Warehousing and Mining **13**(1), 35 (2017)
28. Sanprasit, N., Jampachaisri, K., Titijaroonroj, T., Kesorn, K.: Intelligent approach to automated star-schema construction using a knowledge base. Expert Systems with Applications **182**, 115 – 226 (2021)
29. Sen, P.C., Hajra, M., Ghosh, M.: Supervised classification algorithms in machine learning: A survey and review. In: Emerging Technology in Modelling and Graphics. pp. 99–111 (2020)
30. Tryfona, N., Busborg, F., Borch Christiansen, J.G.: starer: A conceptual model for data warehouse design. In: Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP. pp. 3–8 (1999)
31. Wang, Z., Dong, H., Jia, R., Li, J., Fu, Z., Han, S., Zhang, D.: Tuta: Tree-based transformers for generally structured table pre-training. In: 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. p. 1780–1790 (2021)
32. Yang, Y., Darmont, J., Ravat, F., Teste, O.: Automatic Integration Issues of Tabular Data for On-Line Analysis Processing. In: 16e journées EDA Business Intelligence & Big Data (EDA 2020). vol. B-16, pp. 5–18 (2020)

Appendix: Dataset Information

In this appendix, we detail the information about the datasets we use in our experiments. As mentioned in Section 4.1, these datasets come from different

sources: **AFD** [8], **CDC**[9], **CA**[10], **FR**[11], **KG**[12], **NZ**[13], **UK**[14], **US**[15], **WB**[16]. files from **AFD** and **FR** are in French while the others are in English.

Table 3: Number of files by domains

Economy	**Health**	**Government**	**Environment**	**Society**
143	57	80	28	38

Table 4 shows information about each data source and all data sources (**Total**), including the number of files ($\mathbf{N}_f$), the number of numerical columns ($\mathbf{N}_c$), the number of measures ($\mathbf{N}_m$) and the ratio of number of measures by the number of numerical columns ($\mathbf{R}_m$). Figures in brackets are the minimums and maximums. The original datasets and even more information about them can be found in our github.

Table 4: Data source characteristics

	AFD	**CDC**	**CA**	**FR**	**KG**
$\mathbf{N}_f$	7	28	23	30	106
$\mathbf{N}_c$ **(min-max)**	15 (1-4)	100 (1-4)	156 (2-28)	123 (1-38)	394 (1-17)
$\mathbf{N}_m$ **(min-max)**	8 (0-3)	70 (1-6)	113 (0-28)	39 (0-7)	271 (0-10)
$\mathbf{R}_m$**(%)**	53.33	70.00	72.44	31.71	68.78
	NZ	**UK**	**US**	**WB**	**Total**
$\mathbf{N}_f$	22	42	71	17	346
$\mathbf{N}_c$ **(min-max)**	62 (1-13)	137 (1-9)	311 (1-20)	84 (1-18)	1382 (1-38)
$\mathbf{N}_m$ **(min-max)**	8 (0-3)	99 (0-8)	194 (0-18)	63 (0-13)	900 (0-28)
$\mathbf{R}_m$**(%)**	69.35	72.26	62.38	75.00	65.12

Finally, the datasets that we choose contain at least one numerical column and can be used for DW creation. There were files that are used for other specific purpose, e.g., machine learning, which are not suitable to DW creation. They were thus discarded. There were also files with very poor data quality or completely lacking the information to understand the semantic meaning of columns, which made difficult to tell whether a column could be a measure. Such files were also discarded.

[8] `https://opendata.afd.fr`
[9] `https://data.cdc.gov`
[10] `https://open.canada.ca`
[11] `https://www.data.gouv.fr`
[12] `https://www.kaggle.com`
[13] `https://www.stats.govt.nz`
[14] `https://data.gov.uk`
[15] `https://www.data.gov`
[16] `https://data.worldbank.org`