# Dataconda
# A software Framework for Mining Relational Databases

Michele Samorani

Alberta School of Business
University of Alberta
Edmonton, AB, Canada
Email: samorani@ualberta.ca

*Abstract*—**Dataconda is a software program, freely available to academics on www.dataconda.net, which solves classification and regression problems in a relational database, as opposed to a single table. The user selects a class attribute contained in a table of a relational database, and the software builds and selects predictors by exploring the whole database and aggregating information, without any user intervention. For example, Dataconda may find that the best predictor for "customer value" is the amount of money spent by the customer in cheap electronics, even if the user has not built any such attribute. This demo will introduce a brief theoretical background, illustrate how to use Dataconda in sample databases, and show how to easily extend it.**

***Keywords–Relational Data Mining; Data Preparation; Propositionalization; Classification; Regression***

## I. Theoretical Background on Attribute Generation

Preparing a flat mining table from a relational database is a critical task of the data mining process, because it determines both the predictive performance of the statistical model and the discovery of new knowledge. This task is typically performed manually by an analyst, whereas Dataconda aims at performing it automatically. The main advantage of an automatic approach over a manual one is the ability to find unexpected patterns more easily.

As an example, let us consider the entity relationship diagram of Figure 1, which shows a database of *Purchases*, *Products*, and *Clients*. A client makes 0-to-n purchases through time. Each purchase involves only one product, but a product may be sold in 0-to-n purchases. The goal is to find predictors of the binary attribute *Return* (target attribute) of the table *Purchases* (target table), which indicates whether a purchase was later returned to the store.

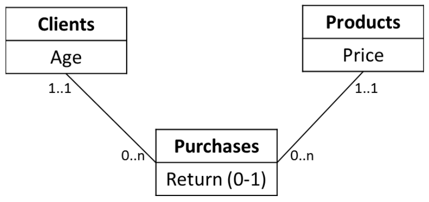


Figure 1. Entity-relationship diagram of a database of purchases

Dataconda adds to the table *Purchases* new attributes that summarize information from other tables, in the hope that they are good predictors of *Return*. The attribute generation procedure is typically referred to as *Propositionalization* [1] and works in two steps: in the first step, the procedure generates a path that starts from the target table; in the second step, a "Roll-up" procedure iteratively joins the tables of the path in order to add a new attribute to the target table, which summarizes the information along the path. The summarization is performed by using refinements, i.e., *where* conditions in the Structured Query Langauge (SQL) or aggregation operators like average (*AVG*), sum (*SUM*), etc. This procedure ultimately results in the addition of a large number of attributes to the target table.

In the example of Figure 1, the first step could generate the path *Purchases* $\rightarrow$ *Clients* $\rightarrow$ *Purchases*. In the second step, the Roll-up procedure will iteratively join the tables of the path in order to generate a new attribute for the table *Purchases*. For example, one attribute that can be generated along this path is the average value of *Return* among the client's past purchases, which represents the client's past return rate.

The initial Propositionalization work in [1] has been improved in [2] with the addition of more aggregation operators; however, the method in [2] does not allow the same table to be traversed twice during step 1 of the procedure, which drastically limits the space of the possible predictors. The method in [3] increases the number of possible refinements; however, their methodology is unsuitable to handle temporal data, because the automatic generation would generate predictors that use future information. These problems are overcome by Dataconda [4].

## II. Using Dataconda

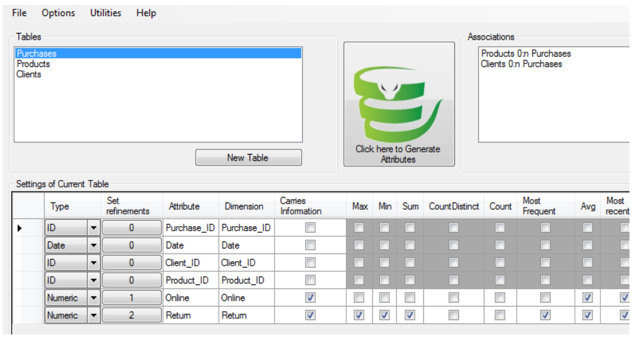
This section gives a brief overview of how to use Dataconda. The software and the complete tutorial [5] can be found online.

### *A. Declare Tables and Relationships*

The first step is to load the tables of an existing relational database and to declare the relationships among them (*0-to-n* or *0-to-1* ). Then, the user needs to define which "aggregating functions" and "refinements" may be used on each attribute.

Figure 2 shows the Dataconda configuration for a database similar to that of Figure 1. Aggregating functions (e.g., *AVG*, *SUM*) are used to aggregate a set of values into one. For example, applying the operator *AVG* on the attribute *Return* enables the generation of an attribute representing the proportion of a client's purchases that were returned.

Refinements, which are the same as "where" conditions in SQL, allow the aggregations to be performed on a subset of rows, as in the attribute *AVG(Return) where Online = 1*, which represents a client's return rate among online purchases.



Purchases
Products
Clients

Products 0:n Purchases
Clients 0:n Purchases

| Type | Set refinements | Attribute | Dimension | Carries Information | Max | Min | Sum | CountDistinct | Count | Most Frequent | Avg | Most recent |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID | 0 | Purchase_ID | Purchase_ID | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| Date | 0 | Date | Date | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| ID | 0 | Client_ID | Client_ID | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| ID | 0 | Product_ID | Product_ID | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| Numeric | 1 | Online | Online | ✓ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ✓ | ✓ |
| Numeric | 2 | Return | Return | ✓ | ✓ | ✓ | ✓ | ☐ | ☐ | ✓ | ✓ | ✓ |

Figure 2. Snapshot of Dataconda

## B. Generate Attributes

The user can then select a *target table* and a *target attribute* in it, and Dataconda will generate all possible predictors for the target attribute that can be obtained using the aggregating functions and refinements defined in the previous step. Practically, the set of columns of the target table is "expanded" with the generated predictors. This process may take seconds or hours, depending on the size of the database and on the number of total predictors to build.

## C. Output of Dataconda

The output of the attribute generation procedure consists of several files, which will be located in the same folder as the data:

- The files *data.csv* and *data.arff*, which contain the flat table (in two different formats);
- A file *attributes.txt* with the English description of each generated attribute;
- A file *Analysis Output.txt*, which contains the detailed result of the attribute selection procedure.

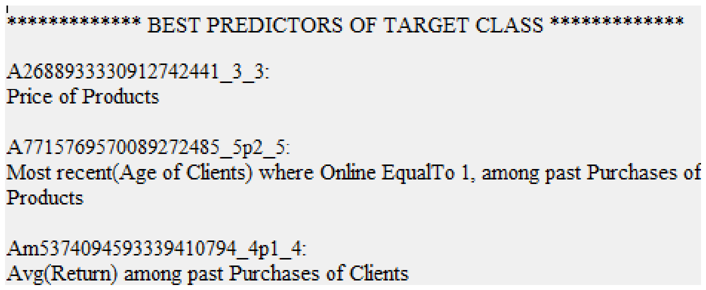
## D. Select Attributes

After generating attributes, Dataconda will find the best predictors through the default attribute selection procedure, which is based on a Lasso regression [6]. Depending on whether the dependent variable is a numeric or a categorical attribute, a linear or logistic Lasso regression is executed with decreasing values of the shrinkage coefficient $\lambda$, so as to retrieve the first 20 attributes that enter the set of selected predictors. Then, these 20 attributes are regressed by themselves against the dependent variable, and only the significant ones (with p-value $\leq 0.05$) are finally returned. Figure 3 shows how the selected predictors are displayed in Dataconda.

The choice of implementing this Lasso-based attribute selection procedure is justified by its computational speed: it takes a time between 0.1 and 1.9 seconds to analyze a data set of about 500 records and 222 attributes. However, this demo will compare the attribute selection and classification performance of several other data mining techniques.

## E. Extending Dataconda

Dataconda can be extended in two ways: by modifying the default attribute selection procedure or by introducing new aggregating operators.



```
************* BEST PREDICTORS OF TARGET CLASS *************

A2688933330912742441_3_3:
Price of Products

A7715769570089272485_5p2_5:
Most recent(Age of Clients) where Online EqualTo 1, among past Purchases of
Products

Am5374094593339410794_4p1_4:
Avg(Return) among past Purchases of Clients
```

Figure 3. Selected predictors

The attribute selection procedure can be modified by changing the file *RTemplate.R*, which contains the attribute selection procedure executed by Dataconda at the end of the attribute generation procedure.

To add new aggregating operators, the user needs to extend the interface *IAggregatingFunction* by specifying the name, description, and logic of the new aggregating operator. Then, the new code needs to be compiled into a *dll*, which then needs to be placed in the same folder as the executable file *Dataconda.exe*. At start-up, Dataconda will load the new operator and will display it in the graphical interface together with the default operators (see Figure 2) .

## III. CONCLUSION

Preparing a flat mining table from a relational database is a critical task of the data mining process, because it determines the predictive performance and the discovery of new knowledge. This task is typically performed manually by an analyst who, guided by domain knowledge, constructs a set of attributes that will hopefully result in a high predictive performance. Aside from being very time consuming, this process is also unlikely to discover unexpected knowledge, as the attributes in the mining table are those that the analyst "suspects" being related to the target attribute.

By automatically generating new attributes, Dataconda can discover new knowledge more easily, and it could also lead to a higher predictive performance than the manual process. To assess the validity of this claim, more experimentation is needed both on simulated and on real-world data sets.

## REFERENCES

[1] A. J. Knobbe, M. De Haas, and A. Siebes, "Propositionalisation and aggregates," in *Principles of Data Mining and Knowledge Discovery*, pp. 277–288, Springer, 2001.

[2] C. Perlich and F. Provost, "Distribution-based aggregation for relational learning with identifier attributes," *Machine Learning*, vol. 62, no. 1-2, pp. 65–105, 2006.

[3] M. Samorani, M. Laguna, R. K. DeLisle, and D. C. Weaver, "A randomized exhaustive propositionalization approach for molecule classification," *INFORMS Journal on Computing*, vol. 23, no. 3, pp. 331–345, 2011.

[4] M. Samorani, "Automatic generation of relational independent variables," in *Proceedings of 2014 INFORMS Workshop on Data Mining and Analytics (DMA 2014) D. Sundaramoorthi, H. Yang, eds.*, 2014.

[5] M. Samorani, *Dataconda User Guide*. http://www.dataconda.net/tutorial.html, 2014. [accessed: 2015-03-03].

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.