

Massively Parallel In-Database Predictions using PMML

Kaushik K Das
EMC/Greenplum

kaushik.das2@emc.com

Eugene Fratkin
EMC/Greenplum

eugene.fratkin@emc.com

Aleksander Gorajek
EMC/Greenplum

aleksander.gorajek@emc.com

Konstantinos Stathatos
Zementis, Inc.

kostas.stathatos@zementis.com

Maulin Gajjar
Zementis, Inc.

maulin.gajjar@zementis.com

ABSTRACT

Like all open standards, the Predictive Model Markup Language (PMML) enables interoperability and portability in the world of data mining and predictive analytics. This means that models developed in any environment and tool set can be deployed and used in a completely different system. Such a level of flexibility creates new opportunities for addressing exceedingly demanding business agility and performance requirements.

One of these requirements is the urgent need to apply the power of predictive analytics to derive reliable predictions—and, hence, business decisions—from vast amounts of data collected by many organizations. In this paper, we discuss how PMML enables embedding advanced predictive models directly into the database or the data warehouse, along side the actual data to be scored. More importantly, we show how we can easily take advantage of highly parallel database architectures to efficiently derive predictions from very large volumes of data.

Categories and Subject Descriptors

H.2.4 [Database Management]: Systems – *parallel databases*.

H.2.8 [Database Management]: Database Applications – *data mining*.

I.5.1 [Pattern Recognition]: Models – *neural nets, statistical*.

General Terms

Standardization, Performance, Languages.

Keywords

Open Standards, Predictive Analytics, Data Mining, PMML, Predictive Model Markup Language, SQL, Parallel Databases, Data Warehouses, MPP, Massively Parallel Processing, Big Data, Collaboration.

1. INTRODUCTION

As advanced analytics becomes pervasive across the enterprise to drive better business decisions, the need for efficient execution of predictive models is paramount. An ever growing array of data mining tools and, all too often, custom specialized

software is used to mine data and derive statistical models from of a wealth of collected data. The ultimate goal is to turn these models into business value by incorporating them into the day to day business operations. This necessitates the ability to integrate them into the IT infrastructure where the outcomes can easily flow into the fingertips of the decision makers. At the same time, the accelerating growth rate of data collected implies that only the most scalable database architectures will be able to meet storage, and more importantly, processing requirements.

In this paper, we discuss how the Predictive Modeling Markup Language (PMML) becomes the bridge between the model development environment and the IT data warehousing infrastructure. We present how, through this open standard, predictive models can be embedded directly in a database, no matter where or how they were created. More importantly, we demonstrate that PMML opens the door for bringing models into a highly parallel database architecture designed for processing data on a massive scale. The result is a data scoring solution that seamlessly integrates advanced predictive analytics with more traditional business analytics, while meeting the volume and performance requirements of the most demanding environments.

We start by providing a brief overview of the three key elements of our study: the PMML standard, the EMC Greenplum Database, and the Zementis Universal PMML Plug-in™, along with an architectural overview of the combined solution. We then discuss the mechanics of moving PMML models from the developer's desktop into the database, mapping models to SQL functions, and using them from SQL statements. Finally, we showcase the scalability of the solution through the results of selected performance experiments.

2. OVERVIEW

EMC Greenplum and Zementis partner to embed the power of PMML into the data warehouse. The joint solution combines the Zementis Universal PMML Plug-in™ for the execution of predictive models with the power and scale of the EMC Greenplum Database, allowing users to score in-place and in-parallel huge amounts of data. Below, we briefly introduce each of the components of the solution, before we present how they work together.

2.1 PMML

As the de-facto standard for data mining models, PMML provides tremendous benefits for business, IT, and the data mining industry in general [1][2][3]. Developed by the Data Mining Group (DMG - <http://www.dmg.org>), an independent, vendor-led consortium, PMML increases business agility by eliminating the need for proprietary solutions or custom code development.

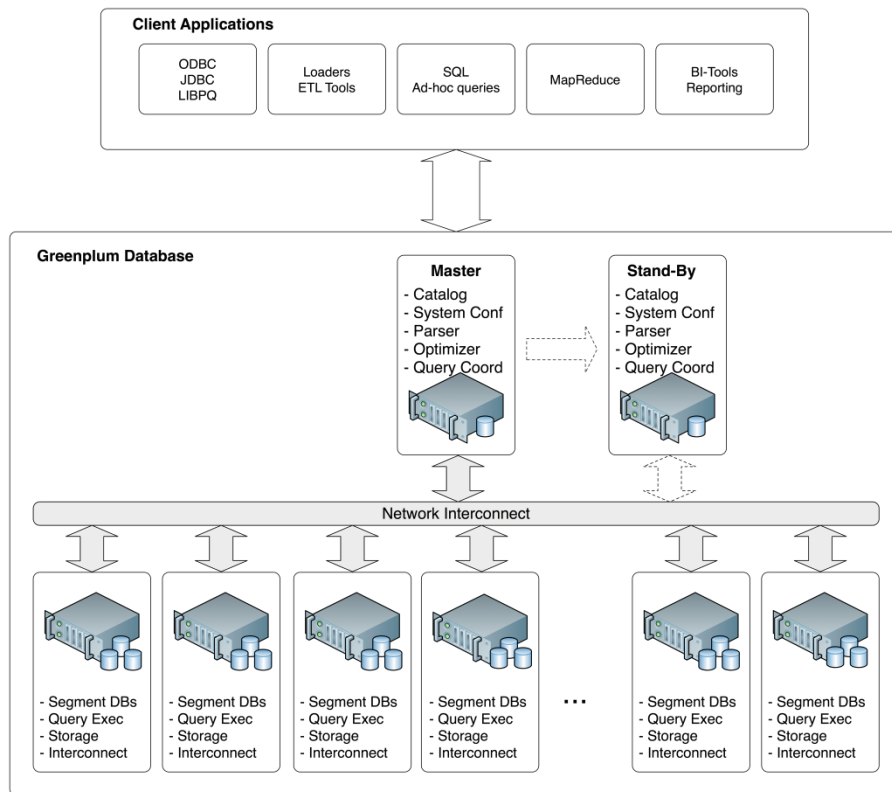


Figure 1. An overview of the shared-nothing, massively parallel EMC Greenplum Database architecture

2.2 EMC Greenplum Database

Today, it is supported by all major data mining tools, commercial and open source. As an open standard, it enables project stakeholders to standardize on one common representation for data mining models. It practically eliminates the barriers and gaps between development and production deployment of predictive analytics. In effect, it minimizes the complexity, cost, and time to turn predictive models into operational IT and business assets.

As the lingua franca for predictive analytics, data mining models can be easily exchanged between PMML-compliant applications. In this way, a model may be built in one statistical tool and easily moved to another for production deployment or visualization. PMML also serves as a bridge between all the teams involved in the data mining process inside a company since it can be used to disseminate knowledge and best practices, thereby stimulating cross-team and inter-organization collaboration. In a world in which sensors and data gathering are becoming more and more pervasive, predictive analytics and standards such as PMML make it possible for organizations to benefit from smart solutions that will truly revolutionize their business.

Besides offering a rich set of structures for describing all the intricate details of a predictive algorithm, PMML also provides information about the input and output of a model. This includes names and types of all input and output data fields, often along with the set of permissible values. In addition, a model expressed in PMML typically includes information about how to handle invalid or missing input values. As we will see later, these elements are essential for the automatic migration of a model into the database and the necessary mappings into the SQL world.

The EMC Greenplum Database is a massively parallel processing (MPP) database, built to support the next generation of data warehousing and large-scale analytics processing. With SQL and MapReduce parallel processing, the database offers industry-leading performance at a low cost for companies managing terabytes to petabytes of data [4][5][6].

The Greenplum Database is an MPP shared-nothing architecture built from commodity hardware components, designed from the ground up to achieve the highest levels of parallelism and efficiency for complex BI and analytical processing. An overview of this architecture is shown in Figure 1. The system distinguishes two types of hosts: (1) a master host and (2) segment hosts. Each segment host is equipped with a dedicated, independent, high-bandwidth channel connection to local disks and it acts as a self-contained database management system that owns and manages its own portion of the overall data. For fault tolerance, there is also usually a stand-by master host as well as mirror stand-by segments for every active segment. The database is also available as a Data Computing Appliance (DCA) which integrates database, compute storage and network into a single, easy-to-manage enterprise-class system.

Besides catalog tables, which are located on the master only, all data is automatically distributed across the segments. The Greenplum Database offers several modes of assigning data to segments. The master accepts incoming connections and after optimizing a statement or query sends a parallel query plan to the segment databases to do the processing. The master does not participate in the processing of queries except for forwarding results to the client as necessary.

```

<DataDictionary numberOfFields="9">
  <DataField name="airtemp" optype="continuous" dataType="double"/>
  <DataField name="buoy" optype="continuous" dataType="double"/>
  <DataField name="day" optype="continuous" dataType="double"/>
  <DataField name="latitude" optype="continuous" dataType="double"/>
  <DataField name="longitude" optype="continuous" dataType="double"/>
  <DataField name="zon_winds" optype="continuous" dataType="double"/>
  <DataField name="mer_winds" optype="continuous" dataType="double"/>
  <DataField name="humidity" optype="continuous" dataType="double"/>
  <DataField name="s_s_temp" optype="continuous" dataType="double"/>
</DataDictionary>
<RegressionModel modelName="ElNino_Linear_Regression_Model"
  functionName="regression" algorithmName="least squares">
  <MiningSchema>
    <MiningField name="airtemp" usageType="predicted"/>
    <MiningField name="buoy" usageType="active"/>
    <MiningField name="day" usageType="active"/>
    <MiningField name="latitude" usageType="active"/>
    <MiningField name="longitude" usageType="active"/>
    <MiningField name="zon_winds" usageType="active"/>
    <MiningField name="mer_winds" usageType="active"/>
    <MiningField name="humidity" usageType="active"/>
    <MiningField name="s_s_temp" usageType="active"/>
  </MiningSchema>
  ...
</RegressionModel>

```

Figure 2. Data dictionary and mining schema for the El Nino regression model

With the data partitioned, query workloads are fully parallelized across all available hardware. Resource contention across servers is avoided and data flows efficiently between segments as query plans dictate. The resulting degree of parallelism and overall scalability far exceeds those of general purpose database systems.

2.3 Universal PMML Plug-in

The Universal PMML Plug-in enables execution of standards-based predictive analytics directly within a database. It shares the PMML execution core with the ADAPA® scoring engine offered by Zementis [7]. It is, however, optimized to be embedded within or close to a database in order to avoid movements of large amounts of data.

In addition, the Plug-in takes on the responsibility of bridging the PMML and SQL world. This means that it presents each loaded PMML model as a SQL function. The name, input parameters and outputs of each function matches the name, input fields, and output fields of the corresponding model as defined in the corresponding PMML file. This way, scoring a data set requires nothing more than writing a SQL statement that involves the SQL functions corresponding to the appropriate models. Predictions (scores, probabilities, categories, clusters, etc.) can be just as easily written back to the database, become part of a report, or passed on to an application.

Like the ADAPA scoring engine, the Universal PMML Plug-in accepts PMML models of all versions (2.0, 2.1, 3.0, 3.1, 3.2 and 4.0) generated by any of the major commercial and open source data mining tools.

2.4 Massively Parallel Predictions

The Universal PMML Plug-in's own shared-nothing design philosophy and replication flexibility is a perfect fit for Greenplum's shared-nothing, MPP architecture. Under the combined solution, each individual segment server houses a separate instance of the Universal PMML Plug-in. PMML models that are loaded into the database are replicated to all segments of the Greenplum installation and are made available for execution.

Data scoring is an inherently parallel operation across the different data segments. This means that queries involving predictive models can be fully parallelized. Segment servers and the Plug-in can take full advantage of their local resources. The net result is the ability to leverage the power of standards-based predictive analytics on a massive scale, right where the data resides.

3. MODEL DEPLOYMENT AND EXECUTION

In this section we describe in detail all the steps required to deploy and execute predictive models in the database. The process starts after the predictive models have been created and have been exported in PMML format from the data mining tool. With the PMML files at hand, it only takes three simple steps to generate predictions from the data in the warehouse:

1. **Preparation:** Validate the PMML files and prepare SQL scripts with definitions for the new SQL functions
2. **Installation:** Copy the PMML files into the segment hosts and install the new SQL functions
3. **Execution:** Run appropriate queries involving the new SQL functions

In the following, we examine each step more closely. Throughout the rest of the Section, we will use as a running example a PMML model we have created with R [2]. It is a linear regression model for the well-known *El Nino* data set. This data set contains oceanographic and surface meteorological readings taken from a series of buoys positioned throughout the equatorial Pacific. The data is expected to aid in the understanding and prediction of El Nino/Southern Oscillation (ENSO) cycles [8].

3.1 Preparation

The first step in deploying one or more models into the database is to create the mapping between the PMML file and SQL. The goal is to create an SQL user defined function (UDF) for each loaded model. These functions will essentially become the wrappers through which the actual models are invoked from within a SQL statement.

```

CREATE FUNCTION
    ElNino_LinR(float8, float8, float8, float8, float8, float8, float8, float8)
RETURNS float8
AS ...

```

Figure 3. Generated SQL function definition for El Nino regression model

```

SELECT ID,
    ElNino_LinR(buoy, day, latitude, longitude, zon_winds, mer_winds, humidity, s_s_temp)
    AS airtemp
FROM elnino_input

```

Figure 4. Sample SQL query using the UDF for the El Nino regression model

A preparatory utility is used to first validate the PMML files and, if necessary, convert them to the latest version. If the files are indeed valid, the same utility generates a SQL script with the UDF definitions for the provided models. There will be one CREATE FUNCTION statement for each model.

As discussed earlier, PMML contains all the required information to automate this step. Let us consider one example. Figure 2 shows the Data Dictionary and Mining Schema of our linear regression model. From the *modelName* attribute of the *RegressionModel* element, we see that the name of the model is *ElNino_LinR*. In addition, this model has seven active (input) mining fields and one predicted (output) mining field. All these fields are of type double. This can be easily mapped to a SQL function with seven IN parameters and one OUT parameter, all of the same numeric type. For the Greenplum Database, the preparation utility generates the function definition shown in Figure 3. The name of the new function is derived from the name of the model as listed in the PMML file (*ElNino_LinR*). Also, like the model, this function has seven input parameters of type float8 and returns a simple value of type float8.

With PMML, it is possible for a model to define more than one output field. For example, classification models may output the winning class in one field along with the corresponding probability. Or in clustering modes, the outputs may include the identifier with the lowest affinity along with the computed affinity. The mapping to SQL in these cases becomes a little more involved as it also requires the generation of complex SQL types which are then used as the OUT parameters of the corresponding functions.

3.2 Installation

With the PMML files validated and the SQL script with the UDFs created, the next step is to install both in the database. Running the SQL script adds the new UDFs in the database catalog. At the same time, a standard copy operation is needed to copy the PMML files to all segment hosts of the Greenplum installation. After this, the database optimizer is aware of the new functions, the database runtime engine knows how to invoke the PMML Plug-in when and where needed and how to pass in the appropriate parameters. At the same time, the Plug-in knows how to execute the appropriate model and return the predictions back to the database.

3.3 Execution

With the UDFs in place, getting predictions from data in the database requires nothing more than invoking the appropriate function in a SQL query. The example in Figure 4 shows a query that invokes the model *ElNino_LinR* to predict the air temperature for all the data in the table *elnino_input*.

Obviously, the models can be used just as seamlessly in more complex queries. Examples would include getting multiple scores from more than one model on the same data, computing prediction distributions over classification models, finding customers in the top or bottom percentiles for marketing campaign success or churn probability, etc. At the same time, such predictions can be just as easily integrated into analytics reports from any BI tool that works with SQL.

4. PERFORMANCE

To demonstrate the versatility and flexibility of the combined solution, we executed a number of tests with different models over different database configurations. In total we used ten different models, four for the El Nino data set and six for the audit data set available with Rattle [9]. The models were created in R using different statistical algorithms and exported in PMML [2]. Table 1 lists the names and types of all ten models.

Table 1. Descriptions of test models

Name	Data Set	Type
ElNino_LinR	El Nino	Linear Regression
ElNino_NN		Neural Network
ElNino_Tree		Decision Tree
ElNino_Clus		Clustering
Audit_Clus	Audit	Clustering
Audit_LinR		Linear Regression
Audit_LogR		Logistic Regression
Audit_NN		Neural Network
Audit_SVM		Support Vector Machine
Audit_Tree		Decision Tree

We present results from two sets of experiments. For the first set, we used a single server running the Greenplum software. For the second set, we used a high performance cluster.

4.1 Single Commodity Server

For the first set of performance tests, we utilized a single server running on commodity hardware configured with 1, 2, or 4 segments. For each of the two data sets we created an input table with ten million records each. The test queries were simple selection queries, similar to that in Figure 4. Each time the output of the queries—the predicted values along with the record ID—was written in a new table in the database.

Figure 5 shows the results of this test in terms of throughput, i.e., records scored per second. It contains the numbers for all ten models for the three different database segment configurations

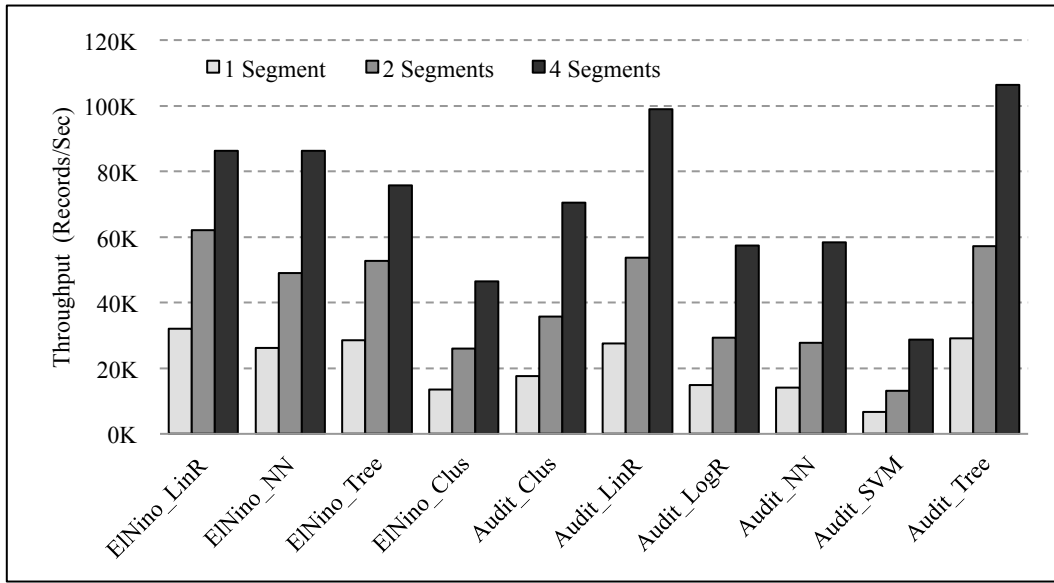


Figure 5. Prediction throughput in records per second for 10 different models on a single commodity server configured with 1, 2, and 4 segments

mentioned above. Note that the time to actually create the output table is included in the computation of the throughput.

If we examine these results, two observations stand out. First, even on commodity hardware the throughput is measured in tens of thousands of records per second. For the simpler models and when using all four available segments, the throughput reaches or exceeds one hundred thousand records per second. In actual time, this corresponds to generating 10 million predictions in less than 100 seconds. This high throughput is the result of the tight integration of the PMML execution engine with the database itself and the ability to score data in-place.

Second, and more importantly, we can see the scalability effects of the shared nothing parallel architecture. In every case, the throughput roughly doubles by doubling the number of segments.

In our experience, this generally holds true as long as the number of segments does not exceed the available hardware resources (CPU cores, memory, I/O bandwidth and channels). Note that the exact scale factor is affected by variables we chose not to explicitly control in these experiments, such as the ratio of data to available buffers per segment.

4.2 High Performance Cluster

For the second set of tests, we utilized a high performance cluster that consisted of two master servers and eight segment servers. The database was configured to use up to 48 segments. We run exactly the same queries, executing the same models as in the first test. Given the significantly higher processing and storage capabilities of the cluster, we also scaled the test up. This time each query scored one billion records to better showcase our claim

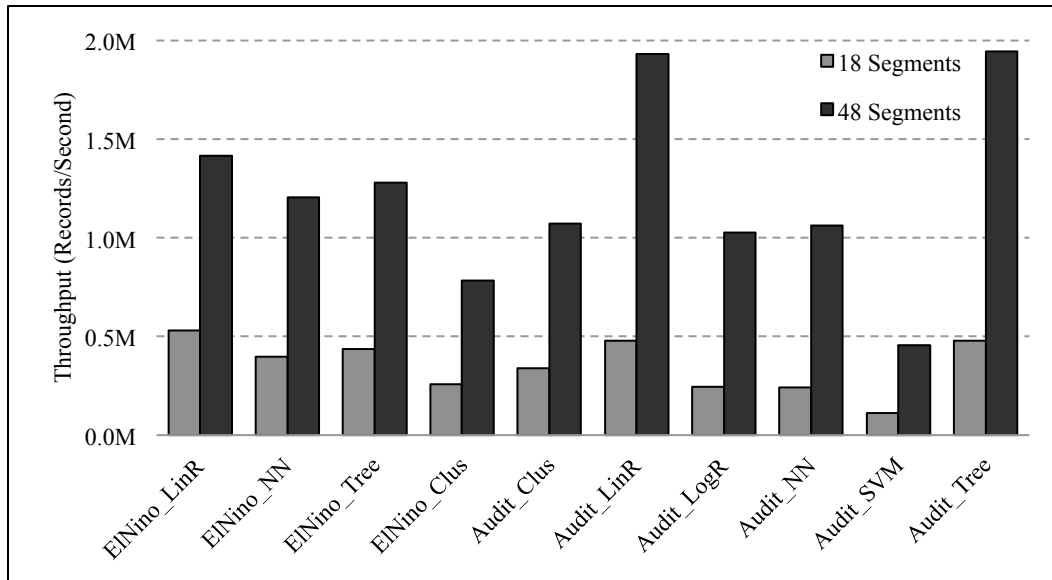


Figure 6. Prediction throughput in records per seconds for 10 different models on a high performance cluster configured with 18 or 48 segments

about computing predictions in a massive scale.

The results of the second test are shown in Figure 6. This time we used two different configurations: one with only 18 segments and one with all 48 segments. What we see from the results is that now the throughput is measured in the order of millions of records per second. In the best case, we achieved a throughput of almost two million records per second. In terms of total time, this means we were able to score one billion records in about nine minutes.

Comparing the two different configurations, we can again see how the performance scales with the number of segments. In fact, for most cases, the relative performance gain is better than the ratio of the number of segments (48/18). This is because, with the data allocated in more segments, each segment processes a smaller amount of data and, therefore, requires a lot less memory paging. On a closer look, we also notice that, with more segments, the tests on the audit data set benefit even more than those on the El Nino data set. This is because the audit data set is bigger than the El Nino in terms of the actual bytes which means, in relative terms, performance improves more when more memory resources are available.

5. CONCLUSION

In Information Technology, open standards are often the catalysts for innovation through portability and interoperability between and across different systems. This is true for PMML, the open standard language for data mining and predictive analytics. With PMML, models can be easily moved from the modeling lab into the IT infrastructure where predictions then drive business processes and automated decisions.

We illustrated that, through the PMML standard, predictive models can be embedded into the database or the data warehouse. Models are automatically turned into SQL functions and can be invoked seamlessly in any SQL statement. In essence this means that the database becomes a feature rich predictive analytics scoring engine, with PMML as a stored procedure language.

In addition, we demonstrated how this approach allows us to take advantage of sophisticated data warehousing solutions that are already available. In particular, we presented how the Zementis Universal PMML Plug-in adds such functionality to the EMC Greenplum Database. Through two separate sets of performance measurements, we illustrated how models execute very efficiently within Greenplum's massively parallel architecture, both on a single server on commodity hardware and on a high performance cluster, achieving a data scoring throughput of up to almost two million predictions (records) per second.

Considering the volume of data collected in warehouses today along with the rising importance of big data, the efficient deployment of predictive analytics becomes all too critical. With PMML as the bridge between the data science lab and the warehouse, we are now able to deliver predictions in a truly massive scale.

6. ACKNOWLEDGMENTS

We would like to thank Alex Guazzelli for providing the models for our performance experiments and Michael Zeller for his suggestions and feedback.

7. REFERENCES

- [1] A. Guazzelli, W. Lin, T. Jena (2010). PMML in Action: Unleashing the Power of Open Standards for Data Mining and Predictive Analytics. CreativeSpace.
- [2] A. Guazzelli, M. Zeller, W. Lin, G. Williams. PMML: An Open Standard for Sharing Models. *The R Journal*, Volume 1/1, May 2009.
- [3] A. Guazzelli (2010). What is PMML? Explore the power of predictive analytics and open standards. *IBM developerWorks*.
- [4] F. Waas. Beyond Conventional Data Warehousing - Massively Parallel Data Processing with Greenplum Database. *In Proc. BIRTE*, 2008.
- [5] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton. MAD Skills: New Analysis Practices for Big Data. *In Proc. VLDB*, 2009.
- [6] F. Waas, J. Kent. Online Expansion of Large-scale Data Warehouses. *VLDB '11*, August 29- September 3, 2011, Seattle, WA
- [7] A. Guazzelli, K. Stathatos, M. Zeller (2009). Efficient Deployment of Predictive Analytics through Open Standards and Cloud Computing. *The ACM SIGKDD Explorations Newsletter*, Volume 11/1, July 2009.
- [8] Hettich, S. and Bay, S. D. (1999). The UCI KDD Archive [<http://kdd.ics.uci.edu>]. Irvine, CA: University of California, Department of Information and Computer Science.
- [9] G. J. Williams. Rattle: A Data Mining GUI for R. *The R Journal*, vol. 1/2, December 2009.