

Java Data Mining API

by

Daniel S. Custodio

Introduction

- Data mining is hot technology
- Gaining ground in finance, manufacturing, marketing, and healthcare
- Becoming widely used because of faster hardware and greater database storage capabilities
- First decade of 21st century would be the decade of data
- Companies will compete with quality and quantity of data
- Even used by Democrats in last Presidential Election

What is data mining?

- “Data mining is a process for finding patterns and relationships in the data, and using that knowledge to classify new data or gain insights into that data.” - Mark Hornick
- A key purpose of data mining is either to help explain the past, or to try to predict the future based on past data.
- Identify patterns in a large volumes of data, and then build models that concisely represent those patterns.
- Data mining is inductive.

Data Relationship

- Data relationship in queries and navigation
- Data relationship with thousands of attributes (high volume and deep complexity of data)

What is JDM 2.0 API

- Java Data Mining – Application Programming Interface
- It's JDBC for data mining
- Developed under the official JSR-247 (Java Specification Request)
- An industry effort to develop a standard Data Mining

Why it was developed?

- Previous Data Mining are proprietary
- Domain of highly skilled specialists
- Migration between data mining tools are difficult
- New data mining tools requires staff retraining
- Avoiding vendor lock-in
- Allows the end-user to adopt different vendor's compliant java data mining systems
- Vendors compete with tool support, pricing, algorithm efficiencies

JDM 2.0 Reference Impl

- No concrete implementations, just stubs or interface

Data Mining Standards

- DMG's Predictive Model Markup Language [DM-PMML]
- OMG's Common Warehouse Metadata for Data Mining [OMG-CWM]
- ISO's SQL/MM Part 6 Data Mining [ISO-SQL/MM]

Data Mining Categories

- Can be classified as **supervised** and **unsupervised**
- **Supervised** predicts a value based on a pre target set (Buy / No Buy, Success / Failure)
Algo: Naïve Bayes for classification
- **Unsupervised** do not need a target and are used to identify intrinsic structure and relations in a body of data. (Customer Segmentation, Naturally occurring groups of proteins)
Algo: K-Means, Apriori Association

Supervise and Unsupervised Learning Algorithms

- Decision Trees
- Neural Networks
- Naïve Bayes
- Support Vector Machines
- K-Means
- Apriori
- Non-negative Matrix Factorization
- ARIMA

Another View

- Descriptive – describes a data set in a concise and summary manner, and presents interesting general properties of data
Algo: K-Means clustering, Decision Tree classification
- Predictive – constructs one or a set of models, performs inference on available dataset, and attempts to predict outcomes for new data sets
Algo: Neural Networks, SVM, K-Means clustering

Data Mining Features

- **Classification** is a type of supervised function where an algorithm builds a model based on a set of predefined predictors used to predict the target. It is usually used in business modeling and credit analysis.
- **Regression** is a type of supervised function. Regression is usually used in financial forecasting, drug response modeling, time series prediction, and environmental modeling.

Data Mining Features ...

- **Clustering** identifies clusters in the data. A cluster is a collection of objects that are similar to each other. Clustering is used primarily in customer segmentation, product groupings, and text mining.
- **Attribute Importance** can be both a supervised and unsupervised function. Attribute importance identifies which attributes are important for building a model.

Data Mining Features ...

- **Association** looks for patterns of relationships in data. This is useful in analyzing consumer behavior, gene and protein analysis, product grouping, customer segmentation, and text mining.
- **Supervised Multi-target** incorporates both classification and regression. Models may include both categorical and numerical targets.

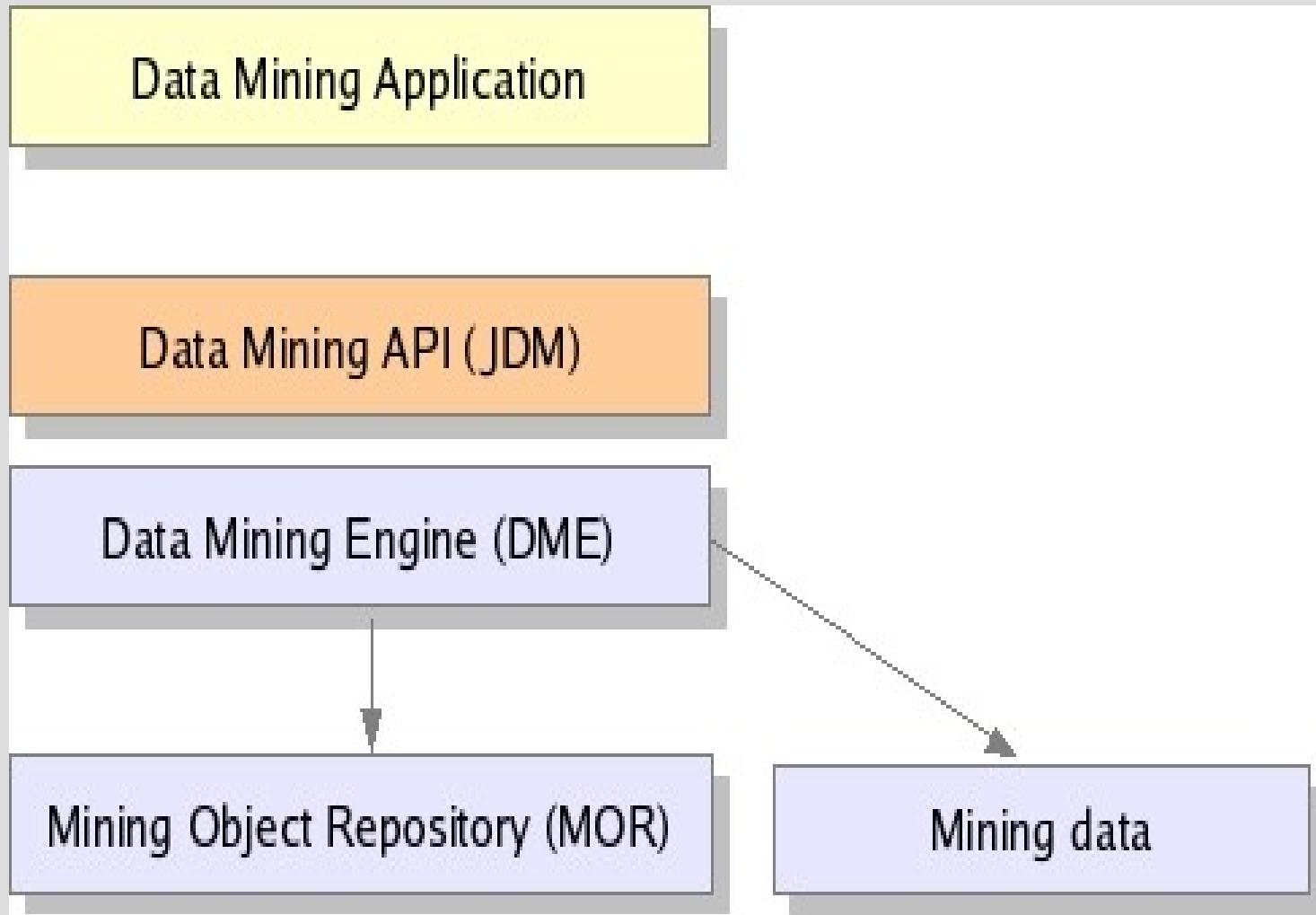
Data Mining Features ...

- **Feature Extraction** creates set of features by decomposing the original data. It is also useful for projecting N dimensions to 2 or 3 dim
- **Time Series** can be used to forecast signal values in the future. It is often used in retail to forecast product demand, but also used in weather prediction, economic indicators, among others.
- **Anomaly Detection** goal is to learn what “normal” is from the vast majority of cases and be able to flag cases that deviate from the norm.

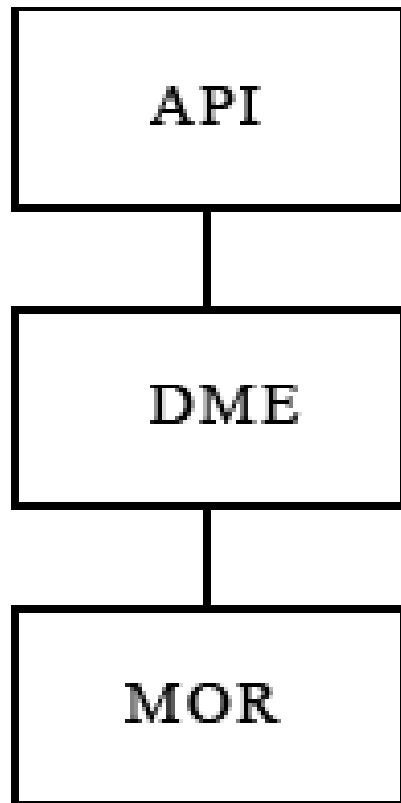
Data Mining Tasks

- Building a model
- Combining the model and data
- Testing the model
- Importing and exporting mining objects and analyzing data.

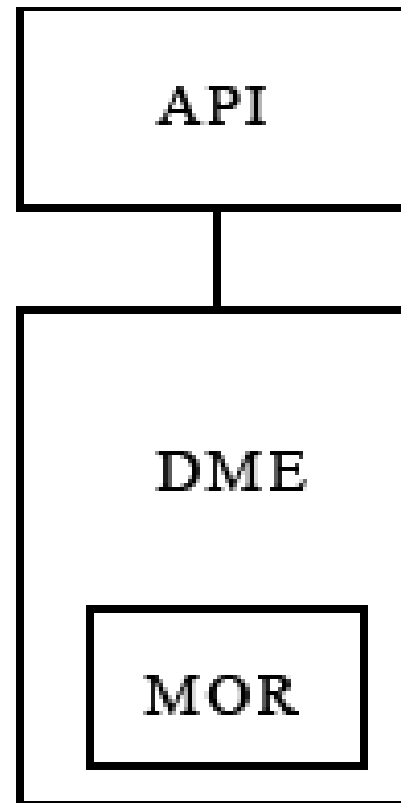
Architectural Components



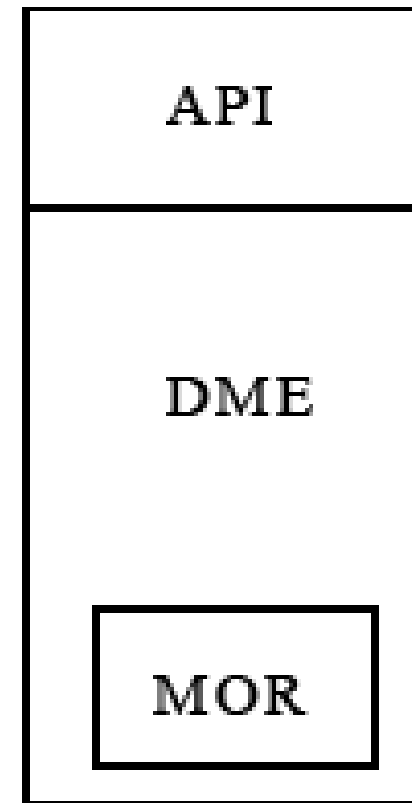
Architecture Options



(a)



(b)



(c)

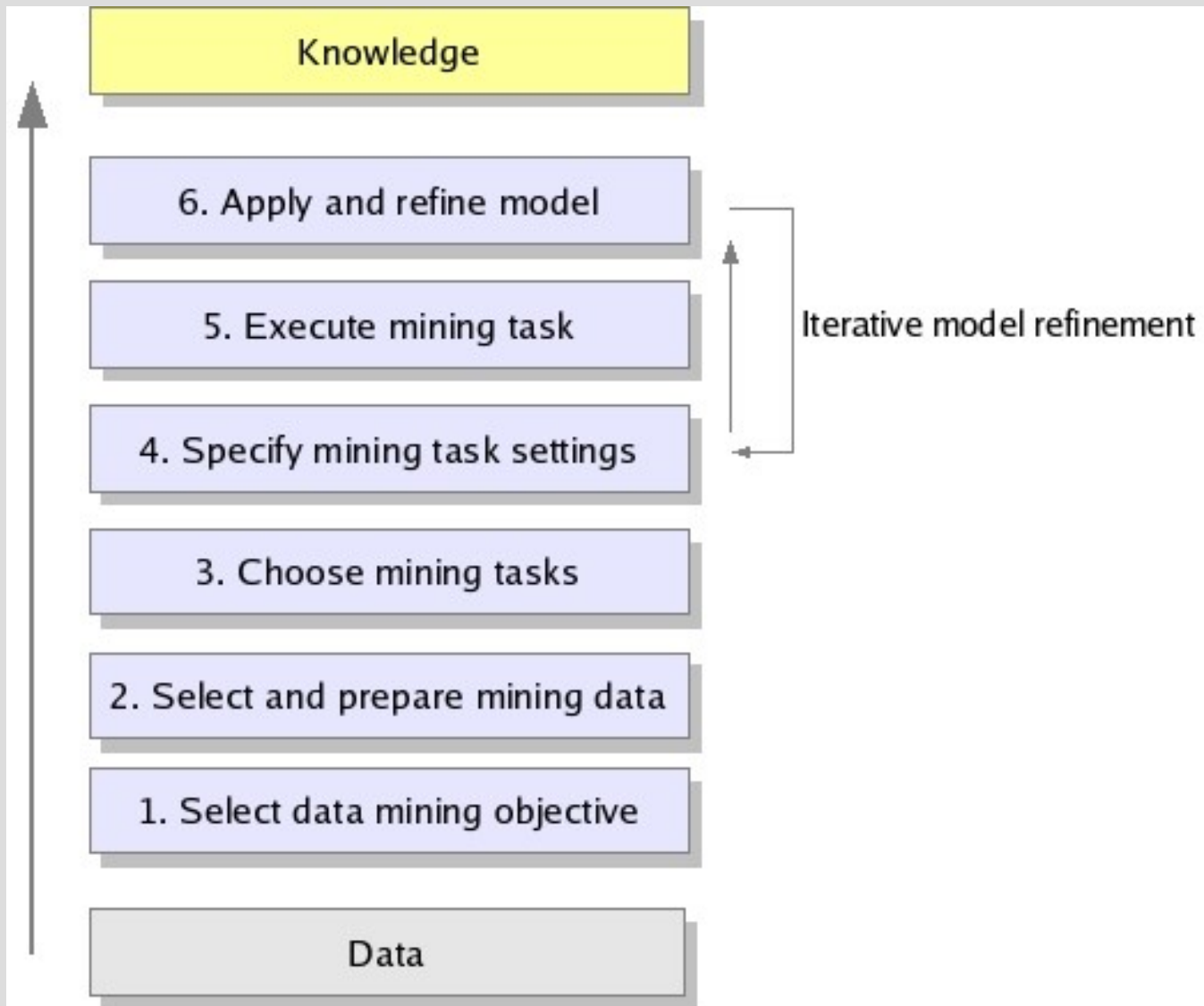
Data Mining Steps

- Identify recurring patterns in data to create a model
- Model can be represented in graphical form, in a set of equations, a neural network, genetic algorithms or even a collection of rules

Data Mining Steps ...

- Decide what you want to learn.
- Select and prepare your data.
- Choose and configure the mining tasks.
- Select and configure the mining algorithms.
- Build your data-mining model.
- Test and refine the models.
- Report findings or predict future outcomes.

Data Mining Steps ...



Data Mining Terms

- **Connection**: Users can get a connection object to access the DME by using a connection factory. The connection object also provides access to the objects present in the metadata repository(MR). Connection objects can create, retrieve and delete mining objects present in the MR.
- **Task**: A task object is used to define tasks that need to be performed by the DME. The JDM defines tasks to build, to apply models and to test. Additionally it provides tasks to compute statistics, importing and exporting data. Tasks can be grouped as batches and can be scheduled for execution by the application.

Data Mining Terms ...

- **Execution Handle and Status**: An asynchronous task produces an execution handle that can be used to track executing tasks or completed tasks. An execution handle also provides a mechanism to block task until it is completed thereby making it a synchronous task. An execution handle can also be used to terminate an executing task, however the actual implementation is left to the vendor who implements the JDM specification.

Data Mining Terms ...

- **Physical Data Set**: Physical data set is the actual data that is being used as input in the data mining operation. The physical data set object can represent relational tables, star schemas, structured files, XML files and olap cubes. In the first release of the specification only table and file data are supported.
- **Physical Data Record**: Physical data records are used for single case scoring both in input and output operations.

Data Mining Terms ...

- **Build Settings**: Build settings are used to set the parameters required for building the model. Build settings allow users to specify the algorithm to be used for building the model. They have default values and they are used if the user omits parameters.
- **Algorithm**: An algorithm is applied to a set of data to produce a model. JDM does not define a large number of algorithms but provides mechanisms to add new ones. An algorithm can optionally have a setting that can be used for setting parameters for an algorithm.

Data Mining Terms ...

- **Algorithm settings**: Algorithm settings are used to set parameters for a particular algorithm that is selected. This helps in fine tuning the algorithm.
- **Model**: A model is produced when an algorithm is applied to a set of data. In the first release models would be read only and will be stored in the metadata repository. A model is specific to the algorithm used to create it and is related to the task that created the model.

Data Mining Terms ...

- **Model Signature**: A model signature defines the input parameters required to use the model. The signature consists of attributes like name, data type, and type.
- **Model Detail**: A model detail object represents the detailed state of a model. The details are specific to the algorithm used and changes when the model or algorithm changes.
- **Attribute Statistics set**: An attribute statistics set contains information about statistics on a set of attributes. It is created when computing statistics are done on a data set object.

Data Mining Terms ...

- **Confusion matrix**: A Confusion matrix is produced when a model is being tested. It tells the user of the model on how well the model is doing in predicting values and where it is making mistakes.
- **Lift**: Lift calculates the ratio between the results computed with and without the predictive model. Lift provides a measure of success of the predictive model.
- **Cost matrix**: A cost matrix defines a tabular representation of the cost associated with predicted values and actual values.

Using JDM API

- The steps to create a data mining model are the following:
 - 1. Identify the data to build your model
 - 2. Specify the type of model to build
 - 3. Create logical representation of data (Optional)
 - 4. Specify parameters to data-mining algorithms (Optional)
 - 5. Create a build task and apply to data references and the build settings

Using JDM API ...

- 6. Verify setting before running build task
- 7. Finally, execute the task.

Using JDM API: Code Example

- **Scenario:** create a model that predicts which customers would purchase a certain product

Using JDM API: Code Example

-
- `// Get data`
- `PhysicalDataSetFactory dataSetFactory`
- `= (PhysicalDataSetFactory)`
`engine.getFactory("javax.datamining.data.PhysicalDataSet");`
- `PhysicalDataSet dataSet =`
- `pdsFactory.create(`
- `"file:///export/data/textFileData.data",`
- `true);`
- `engine.saveObject("buildData", dataSet, false);`

Using JDM API: Code Example

- // Based on physical data, define logical data model
- LogicalDataFactory logicalFactory
- = (LogicalDataFactory)
engine.getFactory("javax.datamining.data.LogicalData");
- LogicalData logicalData = logicalFactory.create(dataSet);
- LogicalAttributeFactory logicalAttributeFactory =
(LogicalAttributeFactory)
- engine.getFactory("javax.datamining.data.LogicalAttribute");
- LogicalAttribute purchase =
logicalData.getAttribute("purchase");
- purchase.setAttributeType(AttributeType.categorical);
- engine.saveObject("logicalData", logicalData, false);

Using JDM API: Code Example

- **// Specify Settings**
- **ClassificationSettingsFactory settingsFactory = (ClassificationSettingsFactory)**
- **engine.getFactory("javax.datamining.supervised.classification.ClassificationSettings");**
- **ClassificationSettings settings = settingsFactory.create();**
- **settings.setTargetAttributeName("purchase");**
- **NaiveBayesSettingsFactory bayesianFactory = (NaiveBayesSettingsFactory)**
- **engine.getFactory("javax.datamining.algorithm.naivebayes.NaiveBayesSettings");**
- **NaiveBayesSettings bayesSettings = bayesianFactory.create();**
- **bayesSettings.setSingletonThreshold(.01);**
- **bayesSettings.setPairwiseThreshold(.01);**
- **settings.setAlgorithmSettings(bayesSettings);**
- **engine.saveObject("bayesianSettings", settings, false);**

Using JDM API: Code Example

- `// Create a build task`
- `BuildTaskFactory buildTaskFactory =`
- `(BuildTaskFactory) engine.getFactory`
- `("javax.datamining.task.BuildTask");`
- `BuildTask buildTask = buildTaskFactory.create("buildData",`
- `"bayesianSettings", "model");`
- `VerificationReport report = buildTask.verify();`
- `if(report != null) {`
- `ReportType reportType = report.getReportType();`
- `//Handle errors here`
- `}`

Using JDM API: Code Example

- //If no errors, save build task
- engine.saveObject("buildTask", buildTask, false);
-
- //Execute the build task
- ExecutionHandle handle = engine.execute("buildTask");
-
- //This may take a long time. So wait for completion
- handle.waitForCompletion(Integer.MAX_VALUE);
-
- // Finally, access the resulting model
- ExecutionStatus status = handle.getLatestStatus();
- if (ExecutionState.success.equals(status.getState())) {
- ClassificationModel model = (ClassificationModel)
- engine.retrieveObject("model", NamedObject.model);
- }

Successful Criteria for Data Mining

- Domain expertise
- Available data
- How the results will be used

Conclusion

- To provide a unified API for all data mining applications or clients
- To facilitate development of data mining applications
- To enable clients to consolidate data with multiple data sources
- To make data mining a new and useful part of an enterprise Java developer's tool chest.
- To make data mining as ubiquitous as JDBC

References

- Java Data Mining Specification Request 247: Java Data Mining (JDM) 2.0
- Java Data Mining (JSR-73): Status and Overview by Mark F. Hornick
- Mine Your Own Data with the JDM API by Frank Sommers
- The Java Data Mining API by Benoy Jose (Java Boutique)
- Data Mining from Wikipedia, The Free Encyclopedia
- JDM 2.0 Javadoc