PMML Feature

The field of use of Data Mining is hard to narrow and also the representation of the process is very abstract. The ***Predictive Model Markup Language (PMML)*** is a

standard exchange format by DMG for the process itself and the structure of the needed data, so that the compability of different software developer is maximized.

PMML follows a XML-Scheme which can be found on the homepage of DMG. The standard (now in v4.3, state october 2017) has about 700 language elements.

PMML can be enhanced at a lot of (nearly all) elements. These enhancements are added as extensions to child-elements at the appropriate elements. Further information can be found here.

The XSD for an extension provides own elements:

```
    <xs:element name="Extension">
  <xs:complexType>
    <xs:complexContent mixed="true">
      <xs:restriction base="xs:anyType">
        <xs:sequence>
          <xs:any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="extender" type="xs:string" use="optional"/>
        <xs:attribute name="name" type="xs:string" use="optional"/>
        <xs:attribute name="value" type="xs:string" use="optional"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

An example for an extension:

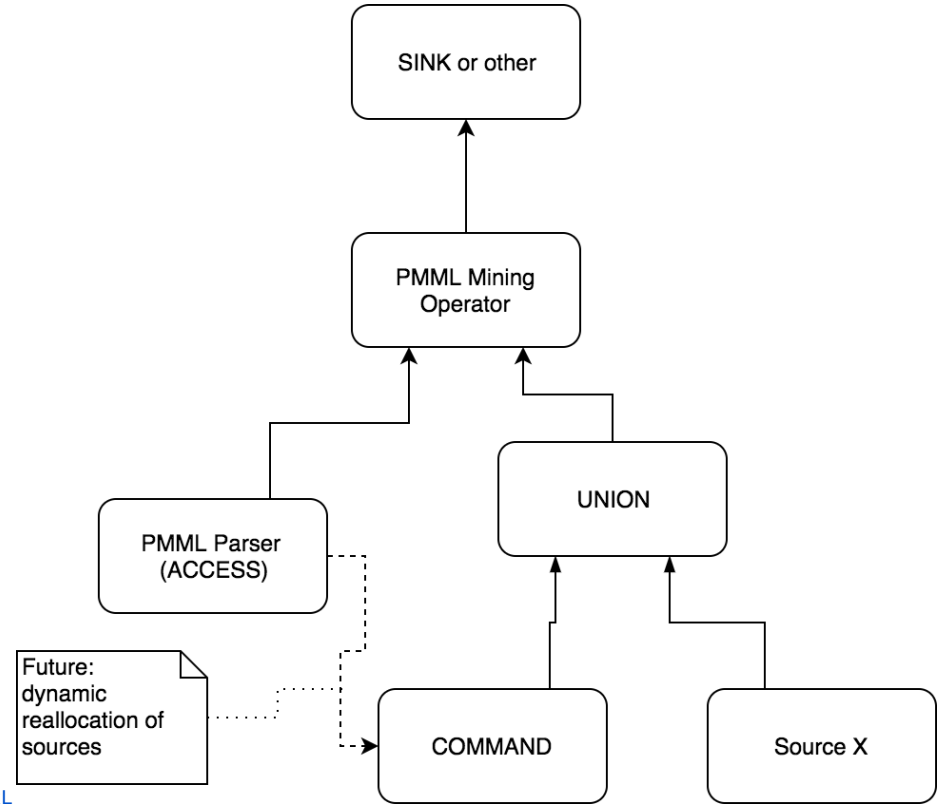
```
    <DataField name="foo" dataType="double" optype="continuous">
  <Extension>
    <DataFieldSource sourceKnown="yes">
      <Source>derivedFromInput</Source>
    </DataFieldSource>
  </Extension>
</DataField>
```

General extensions should be defined for the header. An example for that is, that ***KafkaTopics*** encapsulates data sources for each data schema. To bind to more than one topic, you could create unions with an optional number of topic elements, which serve as data sources.

```
    <Header>
   <Extension>
      <KafkaTopics model="modelA">
         <Topic name="exampleTopic" />
      </KafkaTopics>
      <KafkaTopics model="modelB">
         <Topic name="exampleTopic2" />
         <Topic name="exampleTopicFoo" />
      </KafkaTopics>
   </Extension>
<!-- other Elements -->
</Header>
```

Structure

A possible access architecture might look like this:



blocked URL

The PMML document is loaded by an ***ACCESS*** operator (e.g. by a Kafka topic) and runs through all necessary extensions to get all essential information. A mining operator, which gets the model as an input, can then pass on current models (if a new PMML is send over a topic), without starting the query again. If a ***UNION*** operator is the prefix of a mining operator, you could access further topics over a ***COMMAND*** operator in the future, as long as they don't change during runtime (this type of overloads can be defined in extentions).

PMML Operators

ProtocolHandler:

To access a PMML document you can assign 'PMML' as protocol handler. There are no other parameters necessary and the PMML model has port 0 as an output. You can use e.g. Kafka as a transport handler to reload PMML documents during runtime of queries.

Parameters:

- ***fireOnce***: If the value is set to 'true', then only the very first document is read by the TransportHandler and the state of the operators is set to 'done' afterwards. If the InputStream ends somewhere else, the query is terminated.

Following example shows access to a PMML document provided by DMG.

ProtocolHandler

```
    models = ACCESS({
    source='pmml-data',
    wrapper='GenericPull',
    transport='HTTP',
    protocol='PMML',
    dataHandler='Tuple',
    options=[
        ['uri', 'http://dmg.org/pmml/pmml_examples/rattle_pmml_examples/AuditKMeans.xml'],
        ['method', 'get'],
        ['scheduler.delay', '100000'],
        ['fireOnce', 'true']
    ],
    schema=[
        ['models', 'Object'],
    ]
})
```

Operator:

A PMML document can potentially contain more than one mining models and is therefore outside of the TransportHandler. After the procurement of models you can choose or run a model by name with the operator ***PMML_EVAL***.

The input on port 1 (data) should contain data, which corresponds to the schema of the requested data of the PMML document. More attributes are ignored. Different models can be executed with the same data without integrating more operators in between for filtering, or transporting more meta-data with the analyzed data.

Parameters:

- ***modelName***: Every model within a PMML document has a unique name, so if 'default' is chosen the first model (in XML-order) is executed.
- ***output*** (optional, default: NONE): Describes the output mode of the operator, where you can choose between none, model and input
    - NONE: Output is only the result, the attribute is then called 'prediction'.
    - MODEL: All fields used by the model are output in addition to 'prediction'. 'prediction' is put at the end of the column.
    - INPUT: All values from the data are forwarded to the output.
- ***stackSize*** (optional, default: 100): Describes FIFO cache as a stack, which is used if the data returns elements, but no EvaluationModel was received. As soon as a model is received, all elements in the stack are executed before the next arriving element (until maximum size of the stack is reached). After each new model, the stack gets emptied.

Following example shows usage of the PMML_EVAL operator:

PMML_EVAL Operator

```
    results = PMML_EVAL({
    modelName='KMeans_Model',
    output='input',
    stackSize=5
}, models, data)
```

Complete example:

In pmml_examples you can find examples with correlating data sets:

PMML Example

```
    #PARSER PQL
#RUNQUERY

models = ACCESS({
    source='pmml-data',
    wrapper='GenericPull',
    transport='HTTP',
    protocol='PMML',
    dataHandler='Tuple',
    options=[
        ['uri', 'http://dmg.org/pmml/pmml_examples/rattle_pmml_examples/AuditKMeans.xml'],
        ['method', 'get'],
        ['scheduler.delay', '100000'],
        ['fireOnce', 'true']
    ],
    schema=[
        ['meta', 'Object'],
        ['models', 'Object'],
        ['dictionary', 'Object']
    ]
})

/// data to be tested against
data = ACCESS({
    source='audit-data',
    wrapper='GenericPull',
    transport='HTTP',
    protocol='CSV',
    dataHandler='Tuple',
    options=[
        ['uri', 'http://dmg.org/pmml/pmml_examples/Audit.csv'],
        ['method', 'get'],
        ['readfirstline', 'false'],
        ['delimiter',','],
        ['textDelimiter','"'],
        ['', '']
    ],
    schema=[
        ['ID', 'Integer'],
        ['Age', 'Integer'],
        ['Employment', 'String'],
        ['Education', 'String'],
        ['Marital', 'String'],
        ['Occupation', 'String'],
        ['Income', 'Double'],
        ['Gender', 'String'],
        ['Deductions', 'Double'],
        ['Hours', 'Double'],
        ['IGNORE_Accounts', 'String'],
        ['RISK_Adjustment', 'Double'],
        ['TARGET_Adjusted', 'Double']
    ]
})

results = PMML_EVAL({
    modelName='KMeans_Model',
    output='input',
    stackSize=5
}, models, data)
```