Converting R to PMML

Villu Ruusmann
Openscoring OÜ

"Train once, deploy anywhere"

R challenge

"Fat code around thin model objects":

- Many packages to solve the same problem, many ways of doing and/or representing the same thing
- The unit of reusability is R script
- Model object is tightly coupled to the R script that produced it. Incomplete/split schemas
- No (widely adopted-) pipeline formalization

(J)PMML solution

```
Evaluator evaluator = ...;
List<InputField> argumentFields = evaluator.getInputFields();
List<ResultField> resultFields =
    Lists.union(evaluator.getTargetFields(), evaluator.getOutputFields());
Map<FieldName, ?> arguments = readRecord(argumentFields);
Map<FieldName, ?> result = evaluator.evaluate(arguments);
writeRecord(result, resultFields);
```

- The R script (data pre-and post-processing, model) is represented using standardized PMML data structures
- Well-defined data entry and exit interfaces

Workflow

R challenge

Training:

```
audit_train = read.csv("Audit.csv")
audit_train$HourlyIncome = (audit_train$Income / (audit_train$Hours * 52))
audit.model = model(Adjusted ~ ., data = audit_train)
saveRDS(audit.model, "Audit.rds")
```

Deployment:

```
audit_test = read.csv("Audit.csv")
audit.model = readRDS("Audit.rds")
# "Error in eval(expr, envir, enclos) : object 'HourlyIncome' not found"
predict(audit.model, newdata = audit_test)
```

R solution

Matrix interface (👎):

```
label = audit$Adjusted
features = audit[, !(colnames(audit) %in% c("Adjusted"))]
# Returns a `model` object
audit.model = model(x = features, y = label)
```

Formula interface (👍):

```
# Returns a `model.formula` object
audit.model = model(Adjusted ~ . + I(Income / (Hours * 52)), data = audit)
```

From `pmml` to `r2pmml`

```
~~library("pmml")~~
library("r2pmml")

audit = read.csv("Audit.csv")
audit$Adjusted = as.factor(audit$Adjusted)
audit.glm = glm(Adjusted ~ . + I(Income / (Hours * 52)),
    data = audit, family = "binomial")

~~saveXML(pmml(audit.glm), "Audit.pmml")~~
r2pmml(audit.glm, "Audit.pmml")
```

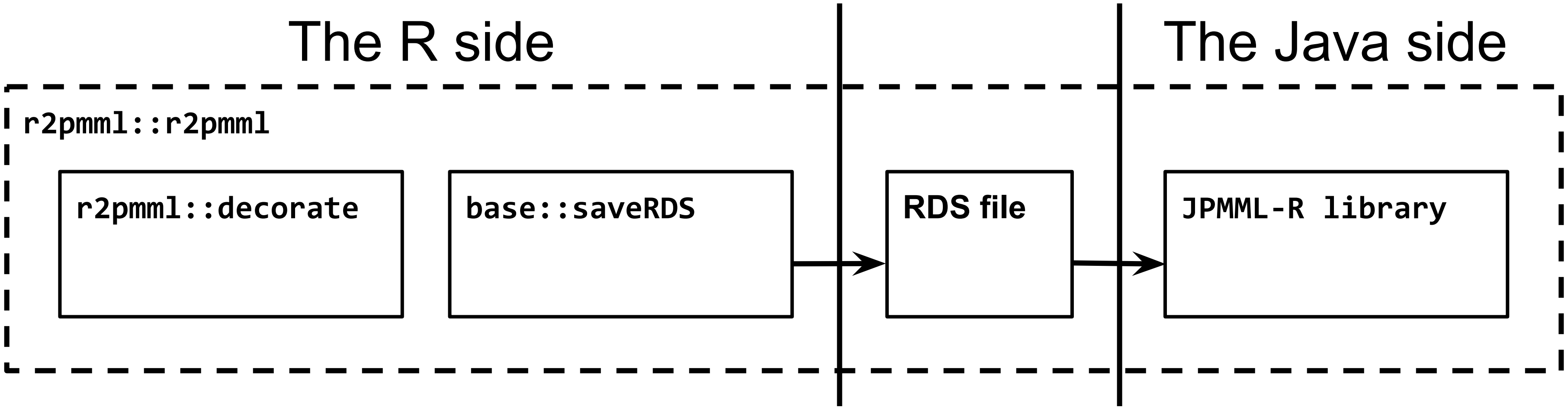
Package `pmml` (👎)

```
<PMML>
  <DataDictionary>
    <DataField name="I(Income/(Hours * 52))" optype="continuous" dataType="double"/>
  </DataDictionary>
  <GeneralRegressionModel>
    <MiningSchema>
      <MiningField name="I(Income/(Hours * 52))"/>
    </MiningSchema>
  </GeneralRegressionModel>
</PMML>
```

Package `r2pmml` (👍)

```xml
<PMML>
  <DataDictionary>
    <DataField name="Income" optype="continuous" dataType="double"/>
    <DataField name="Hours" optype="continuous" dataType="double"/>
  </DataDictionary>
  <TransformationDictionary>
    <DerivedField name="Income/(Hours * 52)" optype="continuous" dataType="double">
      <Apply function="/"><FieldRef field="Income"/><Apply function="*"><FieldRef
field="Hours"/><Constant>52</Constant></Apply></Apply>
    </DerivedField>
  </TransformationDictionary>
  <GeneralRegressionModel>
    <MiningSchema>
      <MiningField name="Income"/>
      <MiningField name="Hours"/>
    </MiningSchema>
  </GeneralRegressionModel>
</PMML>
```

Manual conversion (1/2)



Manual conversion (2/2)

The R side:

```
audit.model = model(Adjusted ~ ., data = audit)
# Decorate the model object with supporting information
audit.model = r2pmml::decorate(audit.model, dataset = audit)
saveRDS(audit.model, "Audit.rds")
```

The Java side:

```
$ git clone https://github.com/jpmml/jpmml-r.git
$ cd jpmml-r; mvn clean package
$ java -jar target/converter-executable-1.2-SNAPSHOT.jar --rds-input
/path/to/Audit.rds --pmml-output /path/to/Audit.pmml
```

In-formula feature engineering

Sanitization (1/2)

```
?base::ifelse

# Static replacement value
as.formula(Adjusted ~ ifelse(is.na(Hours), 0, Hours))

# Dynamic replacement value
as.formula(paste("Adjusted ~ ifelse((is.na(Hours) | Hours < 0 | Hours >
168),", mean(df$Hours), ", Hours)", sep = ""))
```

Sanitization (2/2)

```xml
<DerivedField name="ifelse(is.na(Hours))" optype="continuous" dataType="double">
  <Extension>ifelse(is.na(Hours), 0, Hours)</Extension>
  <Apply function="if">
    <Apply function="isMissing">
      <FieldRef field="Hours"/>
    </Apply>
    <Constant>0</Constant>
    <FieldRef field="Hours"/>
  </Apply>
</DerivedField>
```

Binarization (1/2)

```
?base::ifelse

as.formula(Adjusted ~ ifelse(Hours >= 40, "full-time", "part-time"))
```

Binarization (2/2)

```
<DerivedField name="ifelse(Hours &gt;= 40)" optype="categorical" dataType="string">
  <Extension>ifelse(Hours &gt;= 40, "full-time", "part-time")</Extension>
  <Apply function="if">
    <Apply function="greaterOrEqual">
      <FieldRef field="Hours"/>
      <Constant>40</Constant>
    </Apply>
    <Constant>full-time</Constant>
    <Constant>part-time</Constant>
  </Apply>
</DerivedField>
```

Bucketization (1/2)

```
?base::cut

# Static intervals
as.formula(Adjusted ~ cut(Income, breaks = 3))

# Dynamic intervals
as.formula(Adjusted ~ cut(log10(Income), breaks = quantile(log10(Income))))
```

Bucketization (2/2)

```xml
<DerivedField name="cut(Income)" optype="categorical" dataType="string">
  <Extension>cut(Income, breaks = 3)</Extension>
  <Discretize field="Income">
    <DiscretizeBin binValue="(129,1.61e+05]">
      <Interval closure="openClosed" leftMargin="129.0" rightMargin="161000.0"/>
    </DiscretizeBin>
    <DiscretizeBin binValue="(1.61e+05,3.21e+05]">
      <Interval closure="openClosed" leftMargin="161000.0" rightMargin="321000.0"/>
    </DiscretizeBin>
    <DiscretizeBin binValue="(3.21e+05,4.82e+05]">
      <Interval closure="openClosed" leftMargin="321000.0" rightMargin="482000.0"/>
    </DiscretizeBin>
  </Discretize>
</DerivedField>
```

Mathematical expressions (1/2)

```r
?base::I

as.formula(Adjusted ~ I(floor(Income / (Hours * 52))))
```

Supported constructs:

- Arithmetic operators \`+\`, \`-\`, \`*\`, \`/\`, \`^\` and \`**\`
- Relational operators \`==\`, \`!=\`, \`<\`, \`<=\`, \`>=\` and \`>\`
- Logical operators \`!\`, \`&\` and \`|\`
- Functions \`abs\`, \`ceiling\`, \`exp\`, \`floor\`, \`is.na\`, \`log\`, \`log10\`, \`round\` and \`sqrt\`

Mathematical expressions (2/2)

```
<DerivedField name="floor(Income/(Hours * 52))" optype="continuous" dataType="double">
  <Extension>I(floor(Income/(Hours * 52)))</Extension>
  <Apply function="floor">
    <Apply function="/">
      <FieldRef field="Income"/>
      <Apply function="*">
        <FieldRef field="Hours"/>
        <Constant>52</Constant>
      </Apply>
    </Apply>
  </Apply>
</DerivedField>
```

Renaming and regrouping categories (1/2)

```r
library("plyr")

?plyr::revalue
?plyr::mapvalues

as.formula(Adjusted ~ plyr::revalue(Employment, c("PSFederal" = "Public",
"PSState" = "Public", "PSLocal" = "Public")))
as.formula(Adjusted ~ plyr::revalue(Education, c("Yr1t4" = "Yr1t6",
"Yr5t6" = "Yr1t6", "Yr7t8" = "Yr7t9", "Yr9" = "Yr7t9", "Yr10" = "Yr10t12",
"Yr11" = "Yr10t12", "Yr12" = "Yr10t12")))

as.formula(Adjusted ~ plyr::mapvalues(Gender, c("Male", "Female"), c(0, 1)))
```

Renaming and regrouping categories (2/2)

```
<DerivedField name="revalue(Employment)" optype="categorical" dataType="string">
  <Extension>plyr::revalue(Employment, c(PSFederal = "Public", PSState = "Public",
     PSLocal = "Public"))</Extension>
  <MapValues outputColumn="to">
    <FieldColumnPair field="Employment" column="from"/>
    <InlineTable>
      <row><from>PSFederal</from><to>Public</to></row>
      <row><from>PSState</from><to>Public</to></row>
      <row><from>PSLocal</from><to>Public</to></row>
      <row><from>Consultant</from><to>Consultant</to></row>
      <row><from>Private</from><to>Private</to></row>
      <row><from>SelfEmp</from><to>SelfEmp</to></row>
      <row><from>Volunteer</from><to>Volunteer</to></row>
    </InlineTable>
  </MapValues>
</DerivedField>
```

Interactions

```
as.formula(Adjusted ~
    # Continuous vs. continuous
    Age:Income +
    # Continuous vs. transformed continuous
    Age:I(log10(Income)) +
    # Continuous vs. categorical
    Gender:Income +
    # Categorical vs. categorical
    (Gender + Education) ^ 2
)
```

Estimator fitting

Alternative encodings (1/2)

```
# Continuous label ~ Continuous and categorical features
audit.glm = glm(Income ~ Age + Employment + Education + Marital +
Occupation + Gender + Hours, data = audit, family = "gaussian")

# Encoded as GeneralRegressionModel element
r2pmml(audit.glm, "Audit.pmml")

# Encoded as RegressionModel element
r2pmml(audit.glm, "Audit.pmml", converter = "org.jpmml.rexp.LMConverter")
```

Alternative encodings (2/2)

```
# Binary label ~ Categorical features
audit.glm = glm(Adjusted ~ cut(Age, breaks = c(0, 21, 65, 100)) +
Employment + Education + Marital + Occupation + cut(Income, breaks =
quantile(Income)) + Gender + ifelse(Hours >= 40, "full-time",
"part-time"), data = audit, family = "binomial")

audit.scorecard = r2pmml::as.scorecard(audit.glm)

# Encoded as Scorecard element
r2pmml(audit.scorecard, "Audit.pmml")
```

Q&A

villu@openscoring.io

https://github.com/jpmml
https://github.com/openscoring
https://groups.google.com/forum/#!forum/jpmml

Software (Nov 2017)

- The R side:
  - `base 3.3(.1)`
  - `pmml 1.5(.2)`
  - `r2pmml 0.15(.0)`
  - `plyr 1.8(.4)`
- The Java side:
  - `JPMML-R 1.2(.20)`
  - `JPMML-Evaluator 1.3(.10)`