State of the (J)PMML art

Villu Ruusmann
Openscoring OÜ

Overview

Data Science paradigms

	Structured ("ML")	Unstructured ("AI")
Data	Scalars	Arrays, Sequences
Modeling	Statistics/CPU, Human-driven	Brute force/GPU, "Deep learning"
Audience	Global, from SMEs to mega-corps	SV tech companies (proprietary datasets)
Standards	PMML	ONNX, TensorFlow

Structured information

- Relational databases
  - Schemaful
  - High-level, conceptual features
- Human interpretable algorithms
  - Decision tree
  - Logistic regression
  - Scorecard
  - Business rules
- Machine interpretable algorithms

Training perspective

Structured training workflow

1. Feature engineering
   1.1. Transformation
   1.2. Filtering
2. Modeling
   2.1. Choice of algorithm
   2.2. Hyperparameter tuning
3. Decision engineering
   3.1. Transformation
   3.2. Packaging (following consumer API conventions)

OSS ML frameworks

	R	Scikit-Learn	Apache Spark
Scale	Desktop	Desktop, Server	Server, Cluster
Data model	Rich	Poor	Mixed
Main concept	Model formula	Pipeline	Pipeline
Feature eng.	Average	Very good	Good
Modeling	Very good	Good	Average
Decision eng.	N/A	N/A	Very good

OSS ML algorithm providers

	H2O.ai	XGBoost	LightGBM
Scale	Desktop, Server, Cluster		
Data model	Mixed		
Algorithms	Decision tree ensembles, GLM, Naive Bayes, Neural Networks, Stacking	Decision tree ensembles	Decision tree ensembles

Workflow implementation

- R/Scikit-Learn/Apache Spark everything
- R/Scikit-Learn feature eng. + H2O.ai/XGBoost/LightGBM modeling
- Apache Spark feature eng. + H2O.ai/XGBoost modeling + Apache Spark decision eng.

Mixed workflows bring better results, but are considearbly harder to deploy.

Deployment perspective

The challenge

Productionalization happens on Java, C# or SQL, which have very limited interoperability with R and Python:

- Desktop vs. Server/Cluster
- Language-level threading, memory management issues
- Zero overlap between library sets

ML training and deployment are completely separate concerns and shouldn't be (force-)blended into one.

Possible solutions

- Translation from R/Python application code to Java/C#/SQL application code
- Translation from R/Python application code to some standardized intermediate representation:
  - Higher abstraction level
  - Reorganized, refactored and simplified
  - Stable in time
- Containerization

Deployment options on Java/JVM

- R: N/A
- Scikit-Learn: nok/sklearn-porter
- Apache Spark: Java/Scala APIs, which are tightly coupled to the Apache Spark runtime
- H2O.ai: POJO and MOJO mechanisms
- XGBoost: JNI wrapper, komiya-atsushi/xgboost-predictor-java
- LightGBM: JNI wrapper(?)

The JPMML ecosystem

- "The API is the product"
  - Conversion APIs
  - Analysis and interpretation APIs
  - Scoring APIs
- Layered, role-based design
  - Data scientists, business analysts
  - Data engineers
  - Java/Scala developers
- Automated annotation, quality control

Converting R to PMML

```
library("r2pmml")

audit = read.csv("Audit.csv")
audit$Adjusted = as.factor(audit$Adjusted)

audit.glm = glm(Adjusted ~ . - Age + cut(Age, breaks = c(0, 18, 65, 100))
+ Gender:Education + I(Income / (Hours * 52)), data = audit, family =
"binomial")

audit.glm = r2pmml::verify(audit.glm, audit[sample(nrow(audit), 100), ])

r2pmml::r2pmml(audit.glm, "GLMAudit.pmml")
```

Converting Scikit-Learn to PMML

```
from sklearn2pmml import sklearn2pmml
from sklearn2pmml.pipeline import PMMLPipeline

pipeline = PMMLPipeline([
    ("mapper", DataFrameMapper([...])),
    ("classifier", DecisionTreeClassifier())
])
pipeline.fit(audit_X, audit_y)

pipeline.configure(compact = True, flat = True)
pipeline.verify(audit_X.sample(100))

sklearn2pmml(pipeline, "DecisionTreeAudit.pmml")
```

Scoring PMML

The JPMML-Evaluator® library:

- Small (<1 MB), nearly self-contained (F)OSS Java library
- Developer API consists a single Java interface and a couple of utility classes (model loading, optimization)
- Supports all PMML 3.X and 4.X schema versions
- Vendor extensions:
  - Accessing 3rd party Java libraries in PMML data flow
  - MathML prediction reports

Scoring application scenarios

- Synchronous vs. asynchronous
- Low-latency vs. high-latency
- Individual data records vs. batches of data records
- Embedded vs. over-the-network

Q&A

villu@openscoring.io

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