Open standards for machine learning model deployment
Who is Svetlana Levitan?

Originally from Moscow, Russia

PhD in Applied Mathematics and MS in Computer Science from University of Maryland, College Park

Software Engineer for SPSS Analytic components (2000-2018)

Working on PMML since 2001, ONNX recently

IBM acquired SPSS in 2009

Developer Advocate with IBM Center for Open Data and AI Technologies (since June 2018)

Meetup organizer: Big Data Developers, Open Source Analytics

Two daughters love programming: IIT and Niles North
Agenda

- Deployment challenges
- PMML Internals
- PMML in Python and R
- PMML in IBM products
- PFA
- ONNX
Typical Stages in Machine Learning

1. Collect Data
2. Analyze and Clean Data
3. Transform data
4. Build a Model
5. Deploy the model
6. Monitor and update as needed
Typical Stages in Machine Learning

1. Collect Data
2. Analyze and Clean Data
3. Transform Data
4. Build a Model
5. Deploy the model
6. Monitor and update as needed

(C) 2019 IBM Corp
Model Deployment Challenges

Teams
- Data Scientists and statisticians
- Application developers and IT

Environments
- OS and File Systems
- Databases, desktop, cloud

Languages
- Python or R, various packages, C++ or Java or Scala, Dependencies and versions

Data Preparation
- Aggregation and joins
- Normalization, Category Encoding, Binning, Missing value replacement
DMG to the rescue!

Data Mining Group  dmg.org

Predictive Model Markup Language

• An Open Standard for XML Representation
• Over 30 vendors and organizations
• PMML 4.4 Release manager: Svetlana Levitan
Brief History of PMML versions

0.7 in 1997
First

1.1 in 2000
Six models
Transformations Naïve Bayes, Sequence

2.0 in 2001

3.0 in 2004
Functions
Output Composition SVM, Ruleset

4.0 in 2009
Ensembles,
Cox, Time Series, Model Explanation

4.4 in 2019
More Time Series, Anomaly Detection
Main Components of PMML

- Header
- Data Dictionary
- Transformation Dictionary
- Model(s)
Transformations

- **NormContinuous**: piece-wise linear transform
- **NormDiscrete**: map a categorical field to a set of dummy fields
- **Discretize**: binning
- **MapValues**: map one or more categorical fields into another categorical one
- **Functions**: built-in and user-defined
- **Other transformations**
PMML 4.4 Models

- Anomaly Detection (new)
- Association Rules Model
- Clustering Model
- General Regression
- Naïve Bayes
- Nearest Neighbor Model
- Neural Network
- Regression
- Tree Model

**Mining Model:** composition or ensemble (or both) of models

- Baseline Model
- Bayesian Network
- Gaussian Process
- Ruleset
- Scorecard
- Sequence Model
- Support Vector Machine
- Time Series
Contents of a PMML Model

❖ **Mining Schema:** target and predictors, importance, missing value treatment, invalid value treatment, outlier treatment

❖ **Output:** what to report, post-processing

❖ **Model Stats:** description of input data

❖ **Model Explanation:** model diagnostics, useful for visualization

❖ **Targets:** target category info and prior probabilities

❖ **Local Transformations:** predictor transformations local to the model

❖ **...<Specific model contents>...**

❖ **Model Verification:** expected results for some cases
An example PMML – Data Dictionary, Transformations

```xml
<DataDictionary numberOfFields="5">
  <DataField name="class" optype="categorical" dataType="string">
    <Value value="Iris-setosa"/>
    <Value value="Iris-versicolor"/>
    <Value value="Iris-virginica"/>
  </DataField>
  <DataField name="sepal_length" optype="continuous" dataType="double"/>
  <DataField name="sepal_width" optype="continuous" dataType="double"/>
  <DataField name="petal_length" optype="continuous" dataType="double"/>
  <DataField name="petal_width" optype="continuous" dataType="double"/>
</DataDictionary>

<DerivedField optype="categorical" dataType="double" name="classValue2">
  <NormDiscrete field="class" value="Iris-virginica"/>
</DerivedField>

<DerivedField optype="continuous" dataType="double" name="sepal_lengthNorm">
  <NormContinuous field="sepal_length">
    <LinearNorm orig="4.3" norm="-1.84285714285714"/>
    <LinearNorm orig="7.7" norm="2.3204081632653"/>
  </NormContinuous>
</DerivedField>

<DerivedField optype="continuous" dataType="double" name="sepal_widthNorm">
  <NormContinuous field="sepal_width">
    <LinearNorm orig="2" norm="-2.48539690378995"/>
    <LinearNorm orig="4.4" norm="3.13131926296699"/>
  </NormContinuous>
</DerivedField>
```
Example PMML – Neural Network MiningSchema and inputs

```xml
<NeuralNetwork functionName="classification" activationFunction="tanh">
  <MiningSchema>
    <MiningField name="sepal_length"/>
    <MiningField name="sepal_width"/>
    <MiningField name="petal_length"/>
    <MiningField name="petal_width"/>
    <MiningField name="class" usageType="predicted"/>
  </MiningSchema>
  <NeuralInputs>
    <NeuralInput id="0">
      <DerivedField otype="continuous" dataType="double">
        <FieldRef field="sepal_lengthN20r"/>
      </DerivedField>
    </NeuralInput>
    <NeuralInput id="1">
      <DerivedField otype="continuous" dataType="double">
        <FieldRef field="sepal_widthN20r"/>
      </DerivedField>
    </NeuralInput>
  </NeuralInputs>
</NeuralNetwork>
```

Predictors
Example PMML - Neural Network hidden layer and outputs

```
<Neuron id="6" bias="-0.69138649428932">
  <Con from="0" weight="-0.57324908362227"/>
  <Con from="1" weight="0.892806772564007"/>
  <Con from="2" weight="-1.23192787546061"/>
  <Con from="3" weight="-1.19705013526962"/>
</Neuron>
</NeuralLayer>

<NeuralLayer numberOfNeurons="3" activationFunction="identity" normalizationMethod="softmax">
  <Neuron id="7" bias="0.101922887283541">
    <Con from="4" weight="-1.05690948855012"/>
    <Con from="5" weight="2.00228899161664"/>
    <Con from="6" weight="3.31278374396491"/>
  </Neuron>
  <Neuron id="8" bias="0.917636281284728">
    <Con from="4" weight="-1.47230776836775"/>
    <Con from="5" weight="0.905795272070893"/>
    <Con from="6" weight="-1.60793177845373"/>
  </Neuron>
  <Neuron id="9" bias="-0.2772471777484">
    <Con from="4" weight="2.22290439134024"/>
    <Con from="5" weight="-2.43960637239511"/>
    <Con from="6" weight="-1.3221482019044"/>
  </Neuron>
</NeuralLayer>

<NeuralOutput>
  <DerivedField otype="categorical" dataType="double">
    <FieldRef field="classValue0"/>
  </DerivedField>
</NeuralOutput>
```

- **Hidden layer neuron**
- **Output Layer Neurons**
- **Connecting target to the neurons**
Example PMML for a Tree Model

```xml
<Node id="0"> <True/>
  <Node id="1" score="Iris-setosa" recordCount="50.0">
    <SimplePredicate field="petal_length" operator="lessOrEqual" value="2.6"/>
    <ScoreDistribution value="Iris-setosa" recordCount="50.0"/>
    <ScoreDistribution value="Iris-versicolor" recordCount="0.0"/>
    <ScoreDistribution value="Iris-virginica" recordCount="0.0"/>
  </Node>
  <Node id="2">
    <SimplePredicate field="petal_length" operator="greaterThan" value="2.6"/>
    <Node id="3" score="Iris-versicolor" recordCount="40.0">
      <SimplePredicate field="petal_length" operator="lessOrEqual" value="4.75"/>
    </Node>
  </Node>
</Node>
```
PMML Powered

From
http://dmg.org/pmml/products.html:

Alpine Data
Angoss
BigML
Equifax
Experian
FICO
Fiserv
Frontline Solvers
GDS Link
IBM (Includes SPSS)

JPML
KNIME
KXEN
Liga Data
Microsoft
MicroStrategy
NG Data
Open Data
Opera
Pega
Pervasive Data Rush
Predixion Software
Rapid I

R
Salford Systems (Minitab)
SAND
SAS
Software AG (incl. Zementis)
Spark
Sparkling Logic
Teradata
TIBCO
WEKA

(C) 2019 IBM Corp
**Agenda**

- Challenges
- PMML Internals
- **PMML in Python and R**
- PMML in IBM products
- PFA
- ONNX
PMML in Python

JPMML package is created and maintained by Villu Ruusmann in Estonia.

From https://stackoverflow.com/questions/33221331/export-python-scikit-learn-models-into-pmml

```
pip install git+https://github.com/jpmml/sklearn2pmml.git
```

**Example of how to export a classifier tree to PMML. First grow the tree:**

```python
from sklearn import datasets, tree
iris = datasets.load_iris()
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
```

*SkLearn2PMML conversion takes 2 arguments: an estimator (our clf) and a mapper for preprocessing. Our mapper is pretty basic, since no transformations.*

```python
from sklearn_pandas import DataFrameMapper
default_mapper = DataFrameMapper([(i, None) for i in iris.feature_names + ['Species']])
from sklearn2pmml import sklearn2pmml
sklearn2pmml(estimator=clf, mapper=default_mapper, pmml="IrisClassificationTree.pmml")
```
PMML in R

R packages “pmml” and “pmmlTransformations”

https://cran.r-project.org/package=pmml

Supports a number of R models: ada, amap, arules, caret, clue, data.table, gbm, glmnet, neighbr, nnet, rpart, randomForest, kernlab, e1071, testthat, survival, xgboost, knitr, rmarkdown

Maintained by Dmitriy Bolotov and others from Software AG

JPMML also has package “r2pmml” that augments “pmml” and provides PMML export for additional R models

Build and save a decision tree (C&RT) model predicting Species class:

> irisTree <- rpart(Species~., iris)
> saveXML( pmml(irisTree), "IrisTree.xml" )
Agenda

- Challenges
- PMML Internals
- PMML in Python and R
- PMML in IBM products
- PFA
- ONNX
IBM SPSS Statistics

1968
Statistical Package for Social Sciences

Acquired by IBM in 2009

Subscription option

Integration with Python and R
IBM SPSS Statistics

Transformation PMML from:
ADP (Automatic Data Preparation)
TMS Begin/TMS End

Model PMML from:
COXREG, CSCOXREG
CSGLM, CSLOGISTIC, CSORDINAL
GENLIN, Logistic regression, NOMREG
GENLINMIXED
LINEAR, KNN
MLP, RBF neural networks
NAÏVE BAYES
REGRESSION
TREE, TSMODEL
TWOSTEP CLUSTER

IBM SPSS Modeler

Apriori, CARMA, Association Rules
C5, CART, Chaid decision trees
Cox regression
GENLIN
Decision List
K-Means Cluster
KNN
LINEAR, Regression
Logistic Regression
MLP and RBF
NOMREG
Random Trees
Regression
Two Step Cluster
Score PMML in IBM SPSS

Statistics

Utilities->Scoring Wizard

(C) 2019 IBM Corp
Watson Studio (formerly Data Science Experience)

PMML export possible in Jupyter notebooks, Modeler flows, R Studio. PMML scoring can be done in Flows, notebooks, Watson Machine Learning.
Watson Studio Flows
Scoring PMML in Watson Machine Learning
Benefits of PMML

- Allows seamless deployment and model exchange
- Transparency: human and machine-readable
- Fosters best practices in model building and deployment
Agenda

• Challenges
• PMML Internals
• PMML in Python and R
• PMML in IBM products
• PFA
• ONNX
Portable Format for Analytics - PFA

PMML is great, except when a model or feature is not supported

PFA to overcome this

JSON format, AVRO schemas for data types

A mini functional math language + schema specification

Built-in functions and simple models.

Info: dmg.org/pfa
A Simple Example of PFA (copied from Nick Pentreath’s presentation)

- Example – multi-class logistic regression
- Specify input and output types using Avro schemas
- Specify the action to perform (typically on input)
Known Support for PFA

**Hadrian** (PFA export and scoring engine)  
from Open Data Group (Chicago, IL)

**Aardpfark** (PFA export in SparkML)  
by Nick Pentreath, IBM CODAIST, South Africa

**Woken** (PFA export and validation)  
by Ludovic Claude, CHUV, Lausanne, Switzerland

There was a lot of interest in PFA.

Many opportunities for open source contributions.
Use of PMML and PFA in medical applications

Human Brain Project

Ludovic Claude, CHUV Lausanne, Switzerland
Agenda

• Challenges
• PMML Internals
• PMML in Python and R
• PMML in IBM products
• PFA
• ONNX
ONNX: Open Neural Network eXchange

Since Sep. 2017.  Protobuf
Covers DL and traditional ML
Active work by many companies
ONNX Background

- Initial goal: make it easier to exchange trained models between DL frameworks.
- ONNX github has 20 repos, onnx is the core. Others are tutorials, model zoo, importers and exporters for frameworks.
- Onnx/onnx currently has 12 releases, 112 contributors, 5771 stars.
- Core is in C++ with Python API and tools.
- Supported frameworks: Caffe2, Chainer, Cognitive Toolkit (CNTK), Core ML, MXNet, PyTorch, PaddlePaddle; TF in progress
ONNX use pattern

Frontend
Models in different frameworks

Export

ONNX IR Spec
.onnx

Import

Backend
Models in different frameworks

Tools
Netron visualizer
Net Drawer visualizer
Checker
Shape Inferencer
Graph Optimizer
Opset Version Converter

Training

Inference

PyTorch
Caffe2
Cognitive Toolkit
TensorFlow
mxnet

(© 2019 IBM Corp)
# ONNX tutorials: import and export from frameworks

<table>
<thead>
<tr>
<th>Framework / tool</th>
<th>Installation</th>
<th>Exporting to ONNX (frontend)</th>
<th>Importing ONNX models (backend)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caffe</td>
<td>apple/coremltools and onnx/onnxmltools</td>
<td>Exporting</td>
<td>n/a</td>
</tr>
<tr>
<td>Caffe2</td>
<td>part of caffe2 package</td>
<td>Exporting</td>
<td>Importing</td>
</tr>
<tr>
<td>Chainer</td>
<td>chainer/onnx-chainer</td>
<td>Exporting</td>
<td>coming soon</td>
</tr>
<tr>
<td>Cognitive Toolkit</td>
<td>built-in</td>
<td>Exporting</td>
<td>Importing</td>
</tr>
<tr>
<td>(CNTK)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple CoreML</td>
<td>onnx/onnx-coreml and onnx/onnxmltools</td>
<td>Exporting</td>
<td>Importing</td>
</tr>
<tr>
<td>Keras</td>
<td>onnx/keras-onnx</td>
<td>Exporting</td>
<td>n/a</td>
</tr>
<tr>
<td>LibSVM</td>
<td>onnx/onnxmltools</td>
<td>Exporting</td>
<td>n/a</td>
</tr>
<tr>
<td>LightGBM</td>
<td>onnx/onnxmltools</td>
<td>Exporting</td>
<td>n/a</td>
</tr>
<tr>
<td>MATLAB</td>
<td>onnx converter on matlab central file exchange</td>
<td>Exporting</td>
<td>Importing</td>
</tr>
<tr>
<td>Menoh</td>
<td>pnet-research/menh</td>
<td>n/a</td>
<td>Importing</td>
</tr>
<tr>
<td>ML.NET</td>
<td>built-in</td>
<td>Exporting</td>
<td>Importing</td>
</tr>
<tr>
<td>Apache MXNet</td>
<td>part of mxnet package docs github</td>
<td>Exporting</td>
<td>Importing</td>
</tr>
<tr>
<td>PyTorch</td>
<td>part of pytorch package</td>
<td>Exporting, Extending support</td>
<td>coming soon</td>
</tr>
<tr>
<td>SciKit-Learn</td>
<td>onnx/sklearn-onnx</td>
<td>Exporting</td>
<td>n/a</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>onnx/onnx-tensorflow and onnx/tensorflow-onnx</td>
<td>Exporting - ONNX-Tensorflow</td>
<td>Importing [experimental]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exporting - Tensorflow-ONNX</td>
<td></td>
</tr>
<tr>
<td>TensorRT</td>
<td>onnx/onnx-tensorrt</td>
<td>n/a</td>
<td>Importing</td>
</tr>
</tbody>
</table>
ONNX governance

Steering Committee of 5

**SIGs:**
- Infra
- Operators
- Converters
- Model Zoo

**Working groups:**
- Edge
- Pipelines
- Training
- Testing and compliance
- Foundation
Using ONNX in medical image processing: potential applications

MAX

ibm.biz/model-exchange
Conclusions

Model deployment is an important part of ML lifecycle

DMG works on open standards for model deployment

PMML eases deployment for supported models and data prep

PFA is an emerging standard that needs work

ONNX is becoming a de-facto standard for Deep Learning, needs work!
Links and resources

PMML  dmg.org/pmml
PFA   dmg.org/pfa
ONNX  onnx.ai
CODAIT: codait.org

SPSS:  https://www.ibm.com/analytics/spss-statistics-software
Watson Studio: https://www.ibm.com/cloud/watson-studio
Sign up for free IBM Cloud account:  https://ibm.biz/Bdz8xk

Join Meetup groups: Big Data Developers, Chicago ML

Attend Chicago ML + IBM workshop on Monday, Oct. 28
Thank you.