JavaOne™

# Training and Tracking Machine Learning Models in Java With Tribuo

10/18/2022

JavaOne™

# What is Machine Learning (I)

- Imagine we have an unknown function: `(float[] a) -> float[] b`
- (Supervised) Machine Learning algorithms take many example pairs (a,b) and create an approximate function which when given a produces b
- As simple as an if statement, or as complicated as 100GB of floats
  - Both of them are “machine learning” if the parameters are set automatically based on the data
- Example pairs:
  - An array of pixel values, and the probability of those pixels containing each numerical digit
  - The previous n quarters of sales figures, and the next quarter's sales
  - An array of integers representing words, and an integer representing positive or negative sentiment
- Finding the approximation is called “training a model”, and the approximation itself is the “model”

# What is Machine Learning (II)

- Learned function approximations are different from other kinds of software
- At runtime we have the trained model, but none of the stuff used to train it
- The model behaviour depends on data not present at runtime
  - How close is the data to the ideal specification of the system?
  - Does the data represent the task you want to solve (e.g., in face recognition do you have multiple genders & ethnicities?)
- The model training algorithm usually has many “hyperparameters”
  - Controls the model complexity (e.g., number of if/else statements, number of matrix multiplies)
  - The same model with the same data but different hyperparameters will make different predictions
- Neither of these things are stored in the model itself
  - How do you track systems and debug failures in this environment?

# Using ML Models

- ML models are functions from `float[]` to `float[]`, how do we do something useful?
- The task of running the model given an input to produce the output is called “inference”
- Most of our problems don't look like “take some floats, produce more floats”:
  - “read this table, produce an estimate for next quarter”
  - “tag all the animals in this image”
  - “tag all the people & locations in this text”
- Must convert the data into “features” for the model, and extract outputs from the floats
- ML libraries usually provide the core training & serving functionality
- But getting the data into the floats and parsing the output dimensions is left to the developer
  - Each model needs to be wrapped in supporting code to do all the transformations
  - That code needs to understand which transformations apply to which model
- We're Java developers, it would be nice if the ML tools behaved more like regular code

# Introducing Tribuo (I)

- A Java ML library, developed by Oracle Labs, open sourced in 2020
  - Developed in the open on Github, Apache 2.0 licensed, contributions are welcome under the OCA
  - Website: https://tribuo.org, Github: https://github.com/oracle/tribuo
- Used in production inside Oracle for 4 years
  - Also been integrated into OpenSearch to provide ML functionality there.
- Tribuo 4.x runs on Java 8, we're moving to Java 17 with Tribuo v5
- Our goal is to provide scikit-learn functionality on the JVM
- Implements many ML algorithms, has interfaces to 3rd party ML libraries

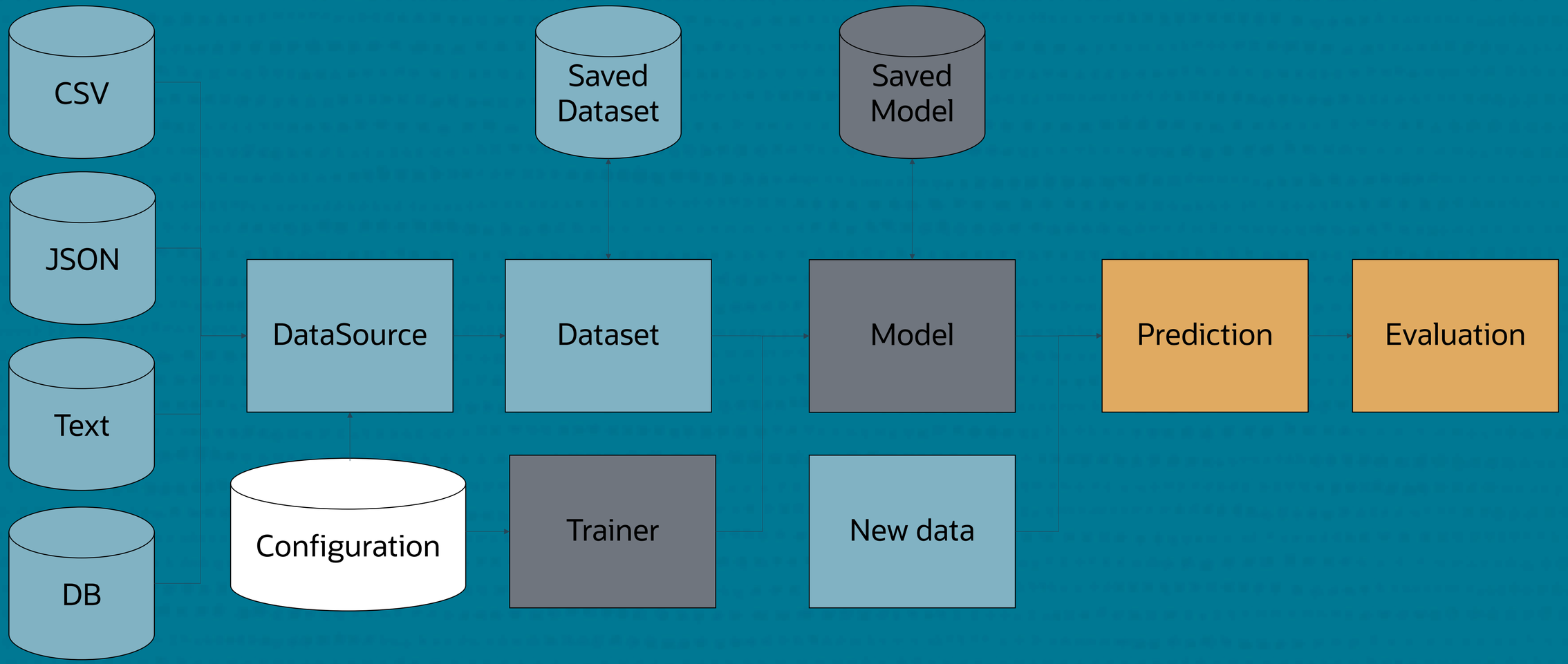
- Tribuo can also serve models trained in other packages or languages
  - ONNX models can be exported from Python packages like scikit-learn, TensorFlow, pytorch etc

# Introducing Tribuo (II)

- Developed to make it easier to deploy ML inside large enterprise applications
  - Like the ones we build at Oracle
- Provides data loading/featurisation required to train models & predict
- Plus evaluation functions to measure the performance of the trained models
- Everything operates on objects, no primitive arrays in user facing methods
  - You can go and poke the floats if you really want to
- Tribuo models accept Example objects and produce Prediction objects
  - Models internally convert the named features from the Example into the float array
  - Then convert the output float array into a named Prediction object

# Tribuo Overview (I)



# Tribuo Overview (II)

- Tribuo covers multiple kinds of ML prediction problems:
  - Anomaly detection, Classification, Clustering, Regression
- We have multiple algorithms for each type of prediction problem
  - Contributions to add new algorithms are welcome
- We have configurable ETL pipelines for columnar and text data.
  - These pipelines are stored inside the model for future use.
- Tribuo models also contain full “provenance” for that model
  - The provenance describes the training algorithm, any hyperparameters, the data distribution, where the data came from, and any transformations applied to that data
  - Tribuo models are “self-describing”, the model knows what it is and where it came from

# Working with features in Tribuo

- Feature spaces in large tabular problems tend to have missing values
- They also usually come from named columns in a structured source
- For that reason all data in Tribuo is implicitly sparse, and features are named
  - The feature name -> vector index mapping is managed by Tribuo
  - Output names are managed in the same way
  - Features which aren't present are ignored in the computation
- Tribuo models ignore new feature names not present in the training data
  - Unseen features have no associated model parameters so they can't affect the output
  - In other systems making the feature vector bigger or smaller can cause issues at runtime
- The number of features used in each prediction is recorded
- Together this helps catch mismatches between training and production

# Provenance in Tribuo (I)

- Provenance is a record of how a Model, Dataset, or Evaluation came to be
- Tribuo's provenance is comprised of two components:
  - Configuration – user controlled parameters for the model or evaluation
  - Instance metadata – metadata from the computation, and optionally user supplied tags
- Provenance is automatic, every Dataset, Model and Evaluation construction creates provenance based on the inputs
- Provenance is pervasive, every relevant object collects provenance
  - Trainers, ETL pipelines, dataset transformations, model ensembles, externally trained models
- It is stored inside the model object, both in memory and on disk
  - So you can't lose it

# Provenance in Tribuo (II)

- Tribuo has a user-facing configuration system, which can configure data sources, trainers and data transformations
  - The system is open-source and was originally developed at Sun Labs in the mid 2000s
  - It supports configuration files in xml, json, edn and protobuf formats
  - Configuration can also be overridden on the command line
- Automatically extracts all configuration from the ETL pipeline and trainers
- ML systems can gather lots of metadata about the model training process
  - The number of features, the number of valid samples, feature distributions
- We also track system level features: OS, CPU architecture, JVM version
  - If available, it stores various file level metadata like hashes and timestamps of the data

# Provenance in Tribuo (III)

- What can we use this pervasive provenance information for?
  - Checking the input pipeline is correct at deployment time
  - Tracking models in production, with self describing models there's no need for external records
  - Reproducing models on new data, or validating that the model can be produced from the data
  - Computing provenance diffs between models to help analyse performance differences
  - Automatic generation of Model Cards, a model description widely used by the ML community
- The last three use cases are external contributions from the Systopia Lab at the University of British Columbia in collaboration with Oracle Labs

- For security, provenance information can be redacted from models

# Training & Provenance Demo

# Working with external models

- Lots of ML training happens on other platforms than the JVM
  - But we still need to deploy those models in Java
- Tribuo supports loading in 3 kinds of models trained by external systems
  - TensorFlow SavedModels, ONNX models and XGBoost models
- They are wrapped in Tribuo Model objects and used like any other Model
  - We also support wrapping OCI Data Science Model Deployment REST endpoints into Tribuo Models
- Allows users to deploy models trained in Python (or other languages) inside their Java programs, using familiar interfaces

# ONNX

- The Open Neural Network eXchange format (ONNX) is a cross-platform model specification format – https://onnx.ai
- Originally developed by Meta & Microsoft to export pytorch and CNTK
- Later support was added for exporting other machine learning models beyond neural networks, like those from scikit-learn
- An open format exported from many ML frameworks, and the models can run on many devices
- With ONNX Runtime Java support we can run models created by many different packages and integrate them into Java applications
  - Our group in Oracle Labs built the ORT Java API and contributed it upstream to ONNX Runtime

# Exporting Tribuo Models in ONNX

- Importing ONNX models is possible with ONNX Runtime, but writing ONNX models from Java is trickier
- The ONNX format is a protobuf, but valid protobufs aren't necessarily valid ONNX models
- Tribuo 4.2 added support for writing ONNX models from Tribuo models
  - Our ONNX writing support is standalone and can be used by other libraries to write ONNX models
- Tribuo Models which implement ONNXExportable can be saved in ONNX
  - Then deployed in Java, Python, C#, node.js or C/C++ via the different ONNX Runtime interfaces

# Model Import and Export Demo

# Conclusions

- We gave an overview of Tribuo, a Java library for training & deploying Machine Learning models
- Tribuo has a strong focus on ease of integration and model tracking
  - Everything operates on objects not primitive arrays
  - Models manage the feature and output indices automatically
  - The extensive provenance make it simple to examine a model and discover how it was created
- Tribuo can also deploy models trained in third-party systems
  - Including models created in other languages like Python
  - Most Tribuo models can be exported in ONNX format for deployment in other systems

# Other Oracle Labs supported ML projects

- Oracle Labs contributes to a number of Java ML projects as well as Tribuo
  - I co-lead TensorFlow-Java - https://github.com/tensorflow/java
  - We develop & maintain the Java interface for ONNX Runtime - https://github.com/microsoft/onnxruntime
  - We contribute to our ML dependencies where appropriate
- We're also open-sourcing a JVM based probabilistic programming language called *Sandwood*
  - Probabilistic programming allows users to specify models and propagate uncertainty through them, leading to better estimates of how likely the outputs are.
  - It'll be available on GitHub soon, under the UPL - https://github.com/oracle/sandwood

# Thank you

Adam Pocock

Principal Member of Technical Staff

Oracle Labs – Machine Learning Research Group

JavaOne™