Apache® Spark™ MLlib 2.0 Preview: Data Science and Production

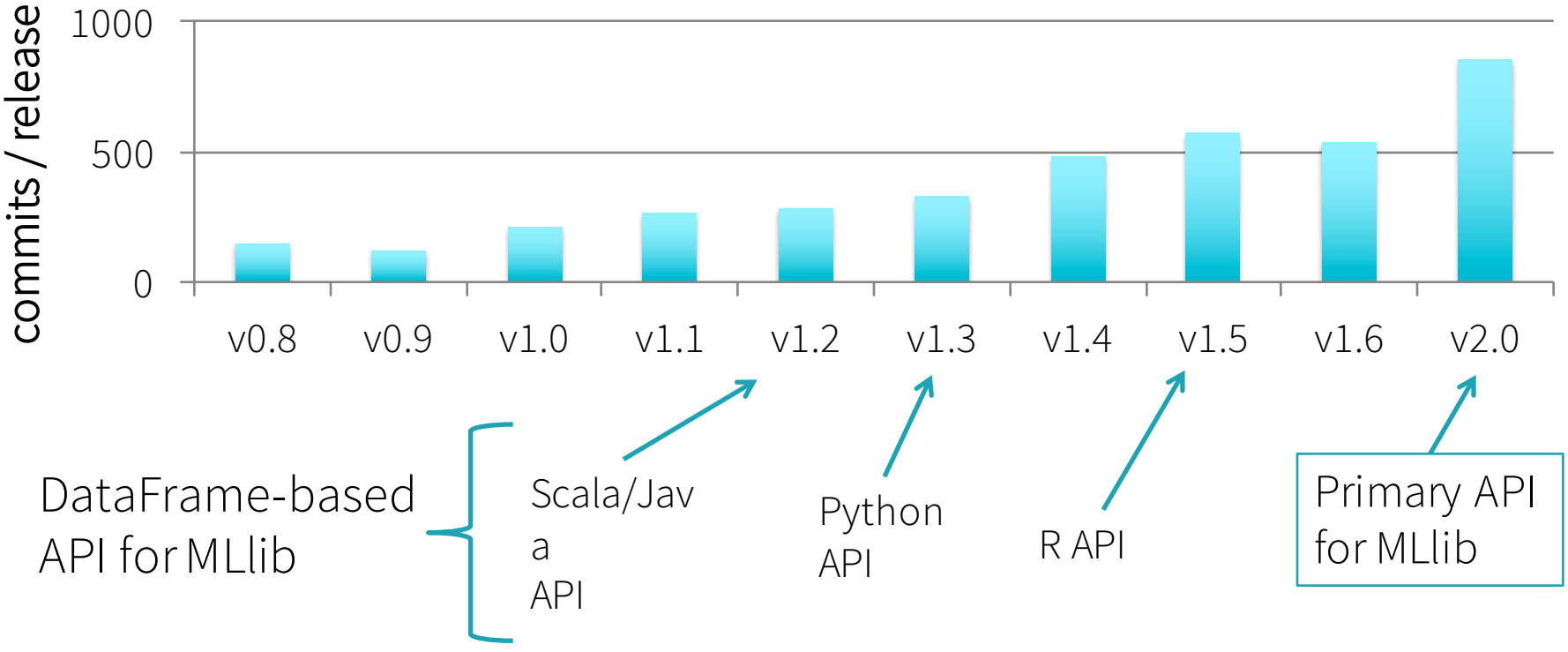
Joseph K. Bradley

June 8, 2016

Who am I?

- Apache Spark committer & PMC member
- Software Engineer @ Databricks
- Ph.D. in Machine Learning from Carnegie Mellon U.

MLlib trajectory



DataFrame-based API for MLlib

a.k.a. “Pipelines” API, with utilities for constructing ML Pipelines

In 2.0, the DataFrame-based API will become the primary API for MLlib.

- Voted by community
- `org.apache.spark.ml, pyspark.ml`

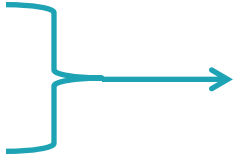
The RDD-based API will enter maintenance mode.

- Still maintained with bug fixes, but no new features
- `org.apache.spark.mllib, pyspark.mllib`

Goals for MLlib in 2.0

Major initiatives

- Generalized Linear Models
- Python & R API expansion



Exploratory analysis

- ML persistence: saving & loading models & Pipelines

Production

Also in 2.0:

- Sketching algorithms: http://databricks.com/blog/2016/05/19
- For details, see SPARK-12626 roadmap JIRA + mailing list discussions.

MLlib for exploratory analysis

- Generalized Linear Models (GLMs)
- Python & R APIs

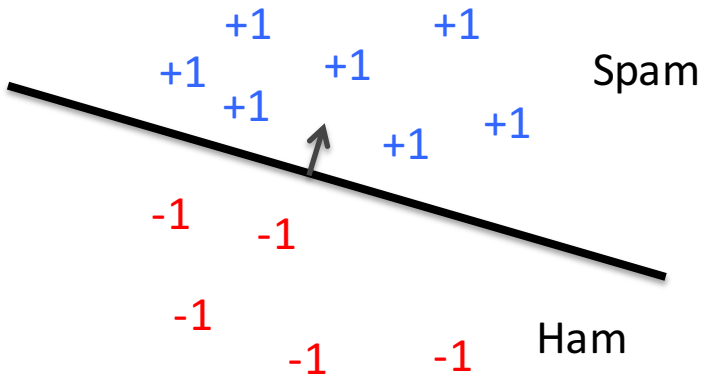
Generalized Linear Models (GLMs)

Arguably *the* most important class of models for ML

- Logistic regression
- Linear regression

→ In Spark 1.6 & earlier

- Many other types of models
- Model summary statistics



GLMs in 2.0

Model family	Supported link functions
Gaussian	Identity, Log, Inverse
Binomial	Logit, Probit, CLogLog
Poisson	Log, Identity, Sqrt
Gamma	Inverse, Identity, Log

Fixes for corner cases

- E.g., handle invalid labels gracefully

`GeneralizedLinearRegression`

- Max 4096 features
- Solved using Iteratively Reweighted Least Squares (IRLS)

`LinearRegression` & `LogisticRegression`

- Millions of features
- Solved using L-BFGS / OWL-QN

Python & R APIs for MLlib

Goal: Expand ML APIs for critical languages for data science

Python

- **Clustering algorithms**: Bisecting K-Means, Gaussian Mixtures, LDA
- **Meta-algorithms**: OneVsRest, TrainValidationSplit
- **GeneralizedLinearRegression**
- **Feature transformers**: ChiSqSelector, MaxAbsScaler, QuantileDiscretizer
- **Model inspection**: summaries for Logistic Regression, Linear Regression, GLMs

R

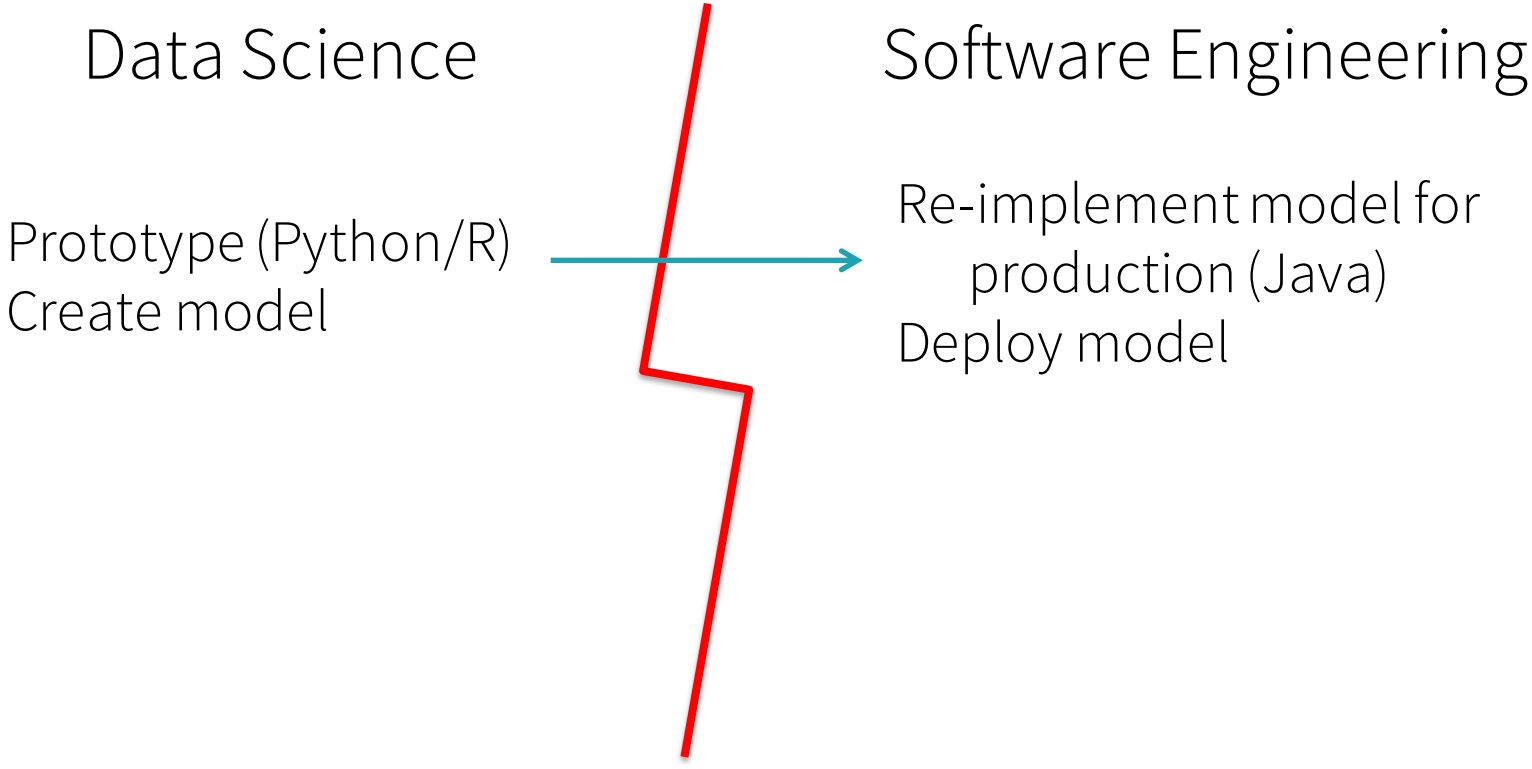
- **Regression & classification**: Generalized Linear Regression, AFT survival regression
- **Clustering**: K-Means

MLlib in production

ML Persistence

Customizing Pipelines

Why ML persistence?

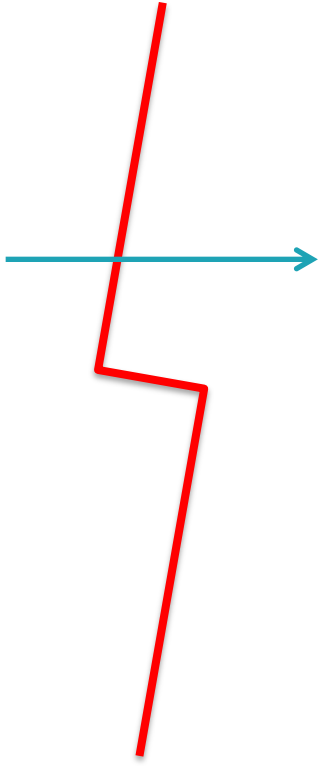


Why ML persistence?

Data Science

Prototype (Python/R)
Create Pipeline

- Extract raw features
- Transform features
- Select key features
- Fit multiple models
- Combine results to make prediction



Software Engineering

Re-implement Pipeline for production (Java)
Deploy Pipeline

- Extra implementation work
- Different code paths
- Synchronization overhead

With ML persistence…

Data Science

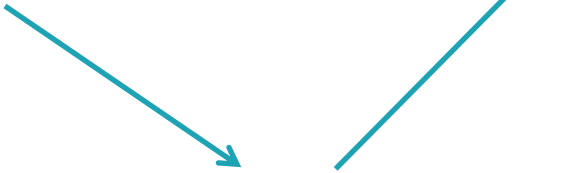
Prototype (Python/R)
Create Pipeline

Software Engineering

Load Pipeline (Scala/Java)

```
Model.load("s3n://…")
```

Deploy in production



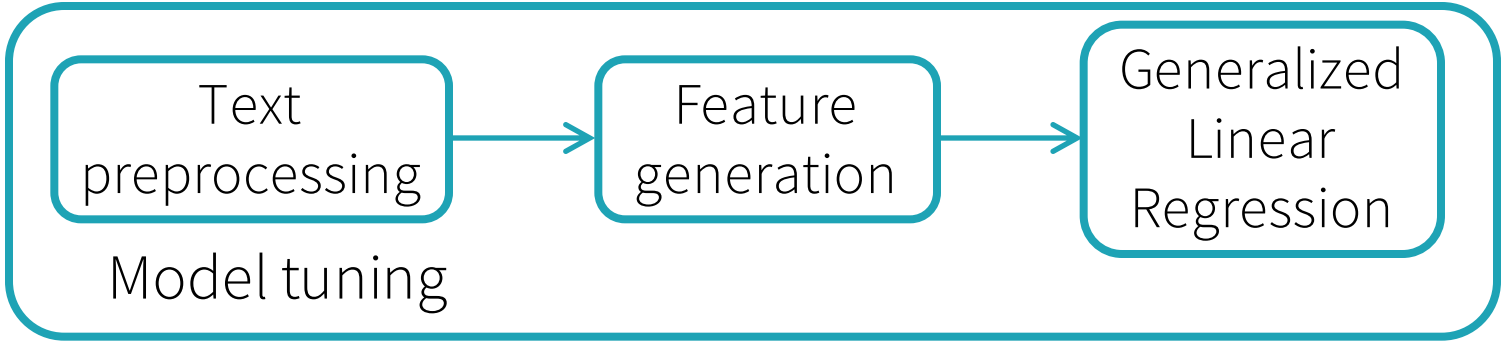
Persist model or Pipeline:

```
model.save("s3n://...")
```

ML persistence status

	"recipe" Unfitted	"result" Fitted
Model	✓	✓
Pipeline	✓	✓

Supported in MLlib's RDD-based API



ML persistence status

Near-complete coverage in all Spark language APIs

- Scala & Java: complete
- Python: complete except for 2 algorithms
- R: complete for existing APIs

Single underlying implementation of models

Exchangeable data format

- JSON for metadata
- Parquet for model data (coefficients, etc.)

Customizing ML Pipelines

MLlib 2.0 will include:

- 29 feature transformers (Tokenizer, Word2Vec, ...)
- 21 models (for classification, regression, clustering, ...)
- Model tuning & evaluation

But some applications require customized Transformers & Models.

Options for customization

Extend abstractions

- `Transformer`
- `Estimator & Model`
- `Evaluator`

Existing use cases:
- Natural Language Processing (Snowball, Stanford NLP)
- Featurization libraries
- Many others!

`UnaryTransformer`: input → output

E.g.:
- `Tokenizer`: text → sequence of words
- `Normalizer`: vector → normalized vector

Simple API which provides:
- DataFrame-based API
- Built-in parameter getters, setters
- Distributed Row-wise transformation

Persistence for customized algorithms

2 traits provide persistence for simple Transformers:

- `DefaultParamsWritable`
- `DefaultParamsReadable`

Simply mix in these traits to provide persistence in Scala/Java.

→ Saves & loads algorithm parameters

```
val myParam: Param[Double]
```

→ Does not save member data

```
val myCoefficients: Vector
```

For an example, see `UnaryTransformerExample.scala` in `spark/examples/`

Recap

Exploratory analysis

- Generalized Linear Regression
- Python & R APIs

Production

- ML persistence
- Customizing Pipelines

Many thanks to the community for contributions & support!

What's next?

Prioritized items on the 2.1 roadmap JIRA (SPARK-15581):

- Critical feature completeness for the DataFrame-based API
  - Multiclass logistic regression
  - Statistics
- Python API parity & R API expansion
- Scaling & speed tuning for key algorithms: trees & ensembles

GraphFrames

- Release for Spark 2.0
- Speed improvements (join elimination, connected components)

Get started

- Get involved via roadmap JIRA (SPARK-15581) + mailing lists
- Download notebook for this talk http://dbricks.co/1UfvAH9
- ML persistence blog post http://databricks.com/blog/2016/05/31

Try out the Apache Spark 2.0 preview release:

http://databricks.com/try

Thank you!

Office hour @ 2:45pm today (Expo Hall)

Twitter: @jkbatcmu