OPEN
DATA
SCIENCE
CONFERENCE

#ODSC

@ODSC

Santa Clara | November 4-6th 2016

Data Science with Spark: Beyond the Basics

Adam Breindel

Instructor: Adam Breindel

LinkedIn: **https://www.linkedin.com/in/adbreind**
Email: adbreind@gmail.com

- 15+ years building systems for startups and large enterprises
- 10 years teaching front- and back-end technology

- Fun big data projects…
  - Streaming neural net + decision tree fraud scoring
  - Realtime & offline analytics for banking
  - Music synchronization and licensing for networked jukeboxes

- Industries
  - Finance
  - Travel
  - Media / Entertainment

Course Objectives

- Use Spark to process dataset features
- Build and evaluate models using common metrics
- Assemble processing, model-building, tuning and evaluation pipelines
- Understand strengths, patterns, mechanisms, and limitations of SparkML
- Perform operations that are not directly exposed in the Spark API
- Extend SparkML with custom feature processing and model algorithms

Not Objectives Today (due to time)

- Core Spark Job Execution, Cluster Configuration
- Data Science Basics, Fundamentals of Predictive Modeling
- "Parade of Algorithms"
  - We won't have lots of tiny examples showing every Spark ML API feature
  - Spark docs and examples folder contains those
- Math and CS of Distributed Algorithms (might cover a very small amount)

Today's Attendees – Spark Experience

This is an exciting workshop to run because there is minimal information ahead of time about attendees ☺

So let's take some informal polls

Approximate Schedule

Part 1 (Morning Session)	**Part 2 (Afternoon)**
Welcome, Intro	NLP, Classification, Clustering
Spark Review, DataFrames and SQL	Model-Parallel Cross-Validation w sklearn
Spark ML Patterns	Integration to External Libraries, Deep Learning
Labs	Extending Spark: Custom Feature Processing
Feature Engineering, Selection	Extending Spark: Custom ML Algorithm

Files and Resources

Documents

- Slides, labs, data available at http://tinyurl.com/odsc-spark

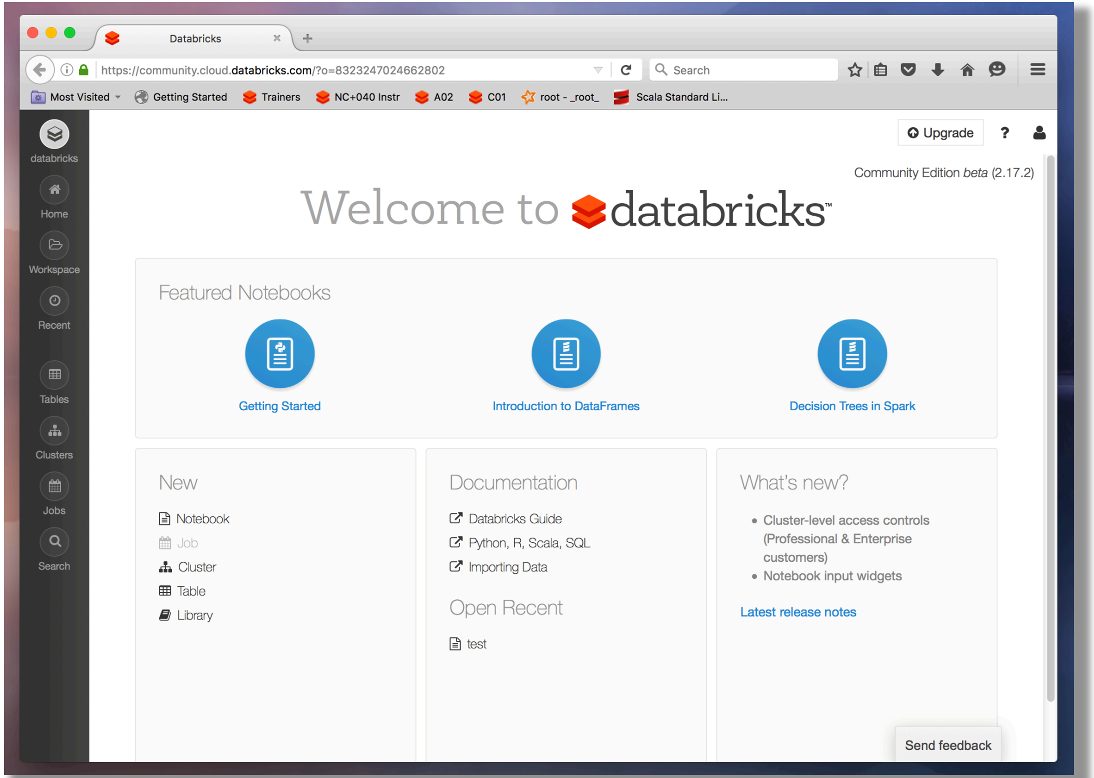
Databricks Account

- Create a Databricks CE account at http://tinyurl.com/databricks-ce
- Use a laptop with Firefox or Chrome (Internet Explorer / MS Edge not supported)

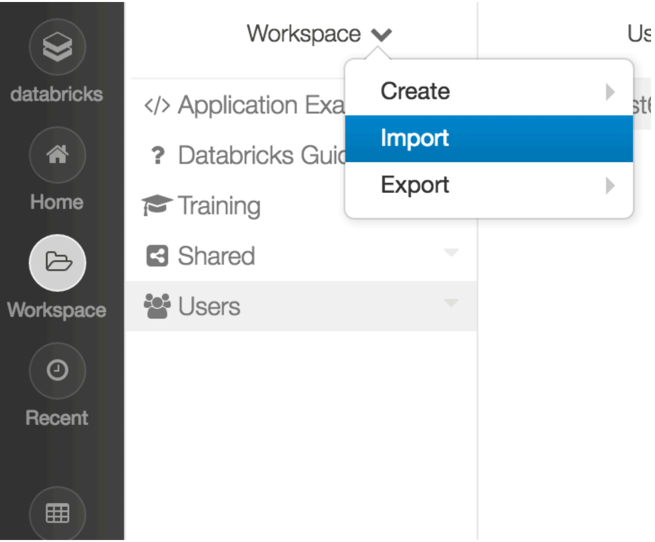
Getting Started

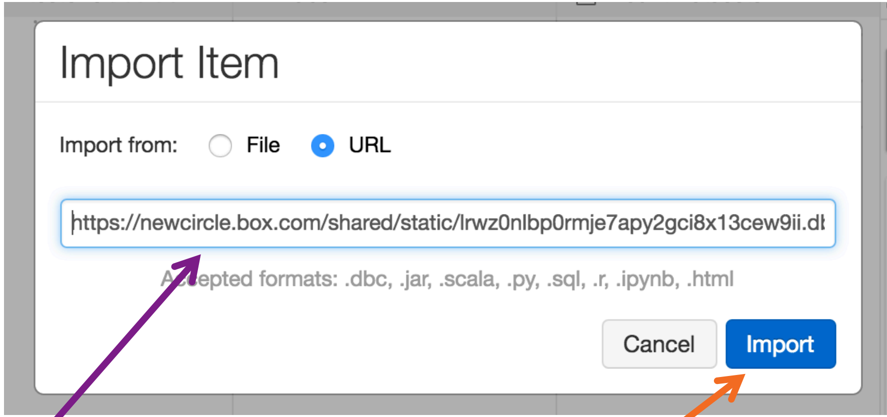
- Get ready type or copy this URL: https://odsc-adbreind.c9users.io/labs.dbc

Log in to Databricks



Import Labs into Databricks

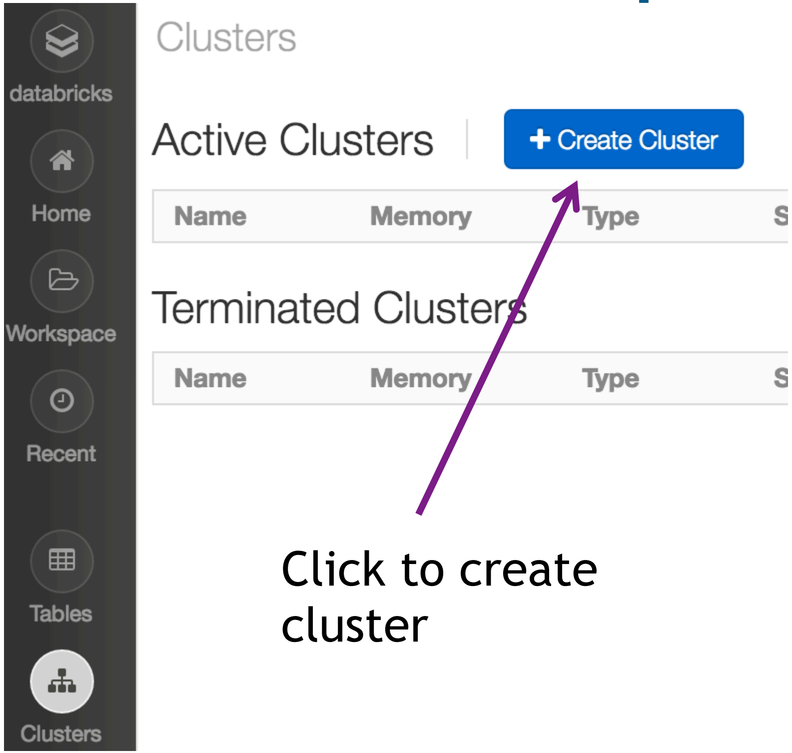




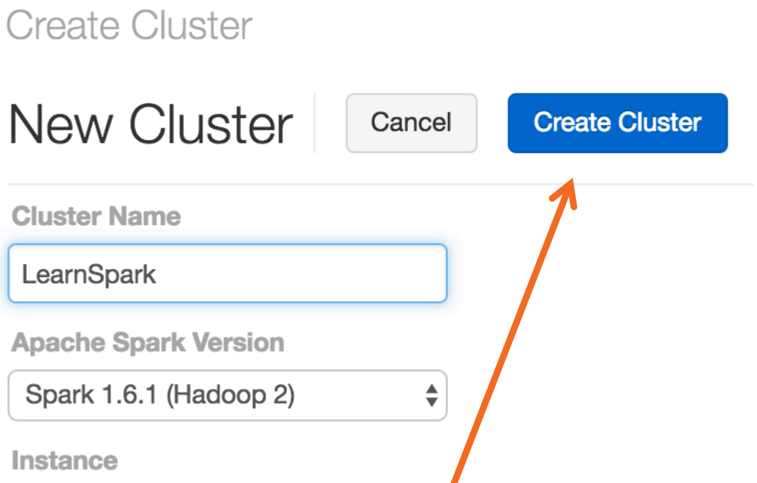
Paste link from previous page

Click import

Create a Spark Cluster!

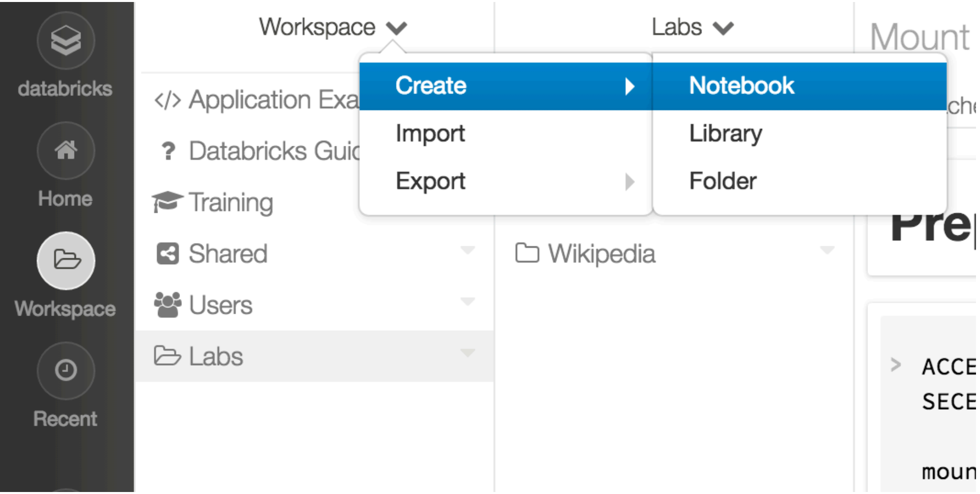


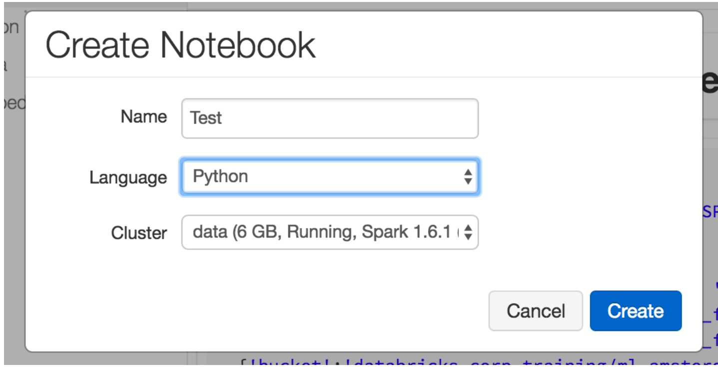
Click to create cluster



- Enter a name
- Choose Spark-2.0.1-db1 (Scala 2.10)
- Click "Create Cluster" to finish

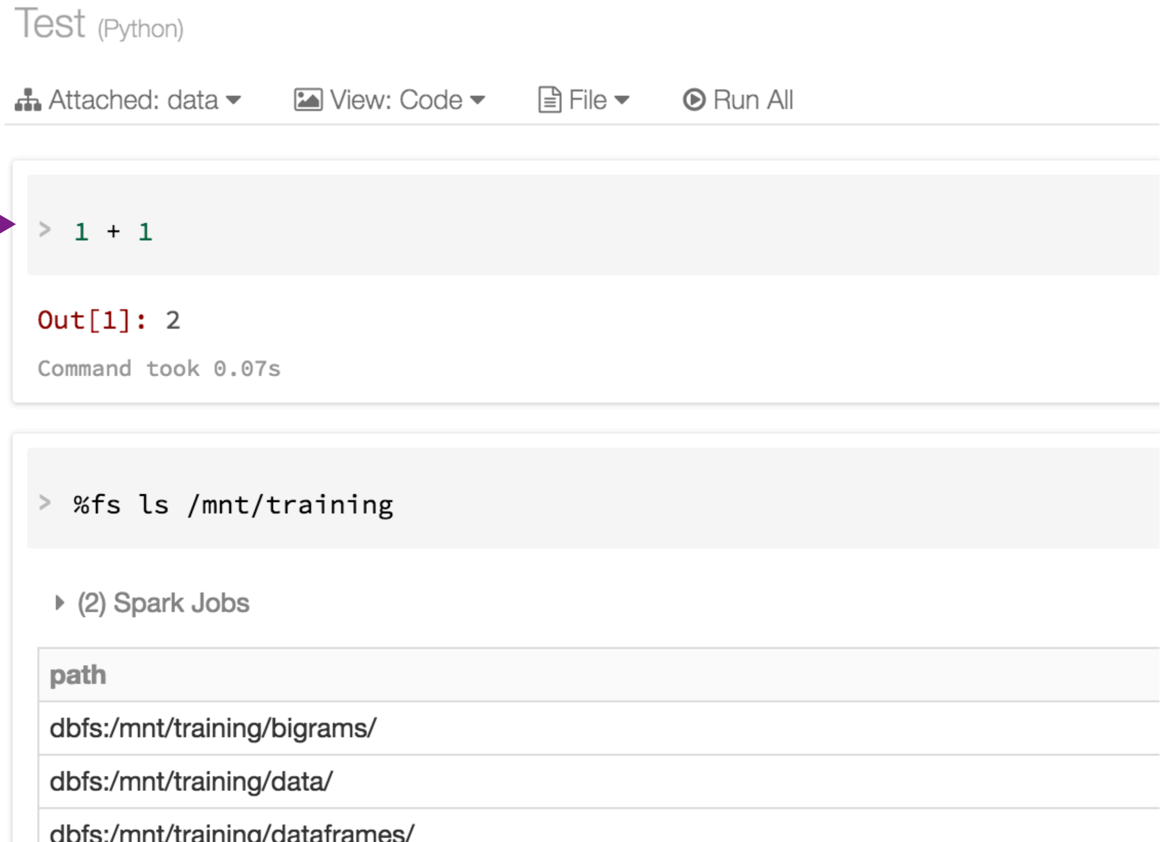
Create a New Notebook…





… and Test

- Key in 1 + 1 and press SHIFT+ENTER to see that the cluster is alive



About Databricks CE

- Create cluster of type `spark-2.0.1-db1 (Scala 2.10)`
- Free, Community Edition for Learning/Exploring APIs
  - Not intended for Machine Learning at Scale
- Limited resources:
  - 6 GB RAM (container)
  - 0.88 cores (vCPU)
  - 8 threads (simulated cores)
  - will shutdown after 60 minutes idle time
    - start a new one … notebooks/DBFS data does persist outside of cluster

About the Data

- Labs feature a variety of real datasets but…
  - Some are "small n" (fewer data points than we might like)
  - Some are "small p" (few predictors / dimensions)
  - Small n / small p can have different mathematical properties from large n / large p
  - Most are relatively clean in format and in terms of content

- If the real world were just like this … it would be easier!
  - (Also we might not need Spark ☹)

What is Spark ?

Spark provides...

- Distributed computation
- Data-parallel architecture
- Fault tolerance
- High scalability
- Data-locality-aware scheduling
- Load balancing

Spark

- Created by Mattei Zaharia at UC Berkeley in 2009
  - Leverage cheap, fast local networking (10GB LAN)
  - Cheap(er) more abundant RAM
  - Less reliance on HDD (though disk still critical)

- Open Source (Apache License)
- Latest Release: v2.0.1 (October 2016)
- Repeatedly Rearchitected for Major (10x+) Perf Gains

Spark ML

Spark for ML: When and Why?

- Use Spark for ML iff you must train and/or predict on data sets too large for single-machine approaches
- Spark has a fairly narrow set of algorithms/models compared to, e.g., R
  - though it is extensible, and many 3rd-party libs are available
- Use Spark exactly when you have to scale out

mllib and spark.ml

mllib = original Spark ML API, based on RDDs

SparkML / MLPipelines / spark.ml = newer API, based on Dataset

- Spark 2.0: RDD-based APIs in spark.mllib have entered maintenance mode.
    - Primary ML API for Spark is now the DataFrame-based API in spark.ml
    - MLlib will still support the RDD-based API in spark.mllib with bug fixes
    - MLlib will not add new features to the RDD-based API

- Spark will add features to DF-based API to reach feature parity with RDD-based API
    - After reaching feature parity (~Spark 2.2), RDD-based API will be deprecated

- The RDD-based API is expected to be removed in Spark 3.0

Why switch to DataFrame-based API?

- DataFrames provide a more user-friendly API than RDDs
- Benefits of DataFrames include
  - Spark Datasources
  - SQL/DataFrame queries
  - Tungsten and Catalyst optimizations
  - Uniform APIs across languages.
- DataFrame-based API provides a uniform API across algorithms and multiple languages.
- DataFrames facilitate practical ML Pipelines, particularly feature transformations

Scala vs. Python

- This class is mostly in Scala... Why?
- We love Python too (and R)! But ... SparkML is built in Scala
  - API is designed to be almost identical in Python
  - but interacting with Scala gives more detailed feedback (e.g., types)
  - and direct access to every public part of the code
- If you extend Spark with new feature processors or models, using the standard pattern, you'll code them in Scala
  - and then maybe wrap them for Python access

Where are the Docs?

- Apache Spark ML Programming Guide site:
  - http://spark.apache.org/docs/latest/ml-guide.html
  - Tons of examples, multiple languages, explanations, links
- API Docs
  - http://spark.apache.org/docs/latest/api/scala/index.html
  - http://spark.apache.org/docs/latest/api/python/pyspark.ml.html
- New! "docs.databricks" site!
  - http://docs.databricks.com/spark/latest/mllib/index.html

SparkML API Patterns

Patterns Underlying Spark ML

"Snap-together brick model"

- Encapsulation of processing
  - Transformer
  - Estimator
  - Pipeline
- Evaluation / Tuning
  - Evaluator
  - CrossValidator
  - ParamGridBuilder

Transformer

- Processes features
- Typically a "map" operation
  - E.g., Binarizer
- But can (occasionally) contain a reduce
  - E.g., OneHotEncoder
- Run by calling aTransformer.transform(aDataframe)
- Extend Spark by extending UnaryTransformer or Transformer

Estimator

- More complex feature processing, and/or model creation
- Typically one or many "reduce" operations
  - Builds (usually expensive and/or large) state
- Produces a Model (Transformer) to encapsulate state
  - Generate state & model by calling anEstimator.fit(aDataFrame)
  - Resulting model is a Transformer
- Extend Spark by creating a specific Model subclass and an Estimator that generates it

Pipeline

- Represents composition of
  - various Transformers' .transform methods
  - various Estimators' .fit and results' .transform methods
- "Point-free" operations
- Is itself an Estimator (supports composition)

Pipeline

Example Goal: rRun tf1 then tf2, then fit/apply est1, and fit est2

Instead of this …

```
model = est2.fit(
                est1.fit(
                     tf2.transform(tf1.transform(data))
                       ).transform(
                     tf2.transform(tf1.transform(data))
                    )
                )
```

We use this …

```
model = Pipeline(stages=[tf1, tf2, est1, est2]).fit(data)
```

Uniform API

- Feature processors: Transformer or Estimator+Model
- ML Algorithms: Estimator
  (> produce >)
- ML Models: Model
- Using any processor, algorithm, and performing tuning uses the same API!
  - Low cognitive load
  - "If you're bored, we're all winning!"
    - (because your brain is now free to work on the interesting hard stuff)

SparkML API Patterns (2)

Evaluator

- Calculates statistics on our models indicating
  - goodness-of-fit, explanation of variance
  - error quantities, precision/recall/etc.
- Generates 1 stat at a time
  - "mode-ful" switching of stat via setter
- Why? Designed for integration and for Spark, not just us
  - In particular, answers question "Which is better?" for tuning
- RegressionEvaluator, BinaryClassificationEvaluator, ...

ParamGridBuilder

- Helper to specify a grid of (hyper)params
  - Several params, chosen based on algorithm/model type
  - Several values for each param
  - Allows Spark to find/try every combination of values!

Parameter	**Test Value 1**	**Test Value 2**	**Test Value 3**	**(etc.)**
maxDepth	6	10	12	
maxBins	16	32	48	
(etc.)				

CrossValidator

- Performs K-fold cross-validation for…
  - combinations of param values from attached Param Grid
  - using attached Estimator (typically, algorithm or Pipeline)
  - and attached Evaluator (indicating better/worse results)

Example: 3-fold cross validation … Divide training set in thirds, run 3 passes with the same data

Pass 1:	Train	Train	Validate
Pass 2:	Train	Validate	Train
Pass 3:	Validate	Train	Train