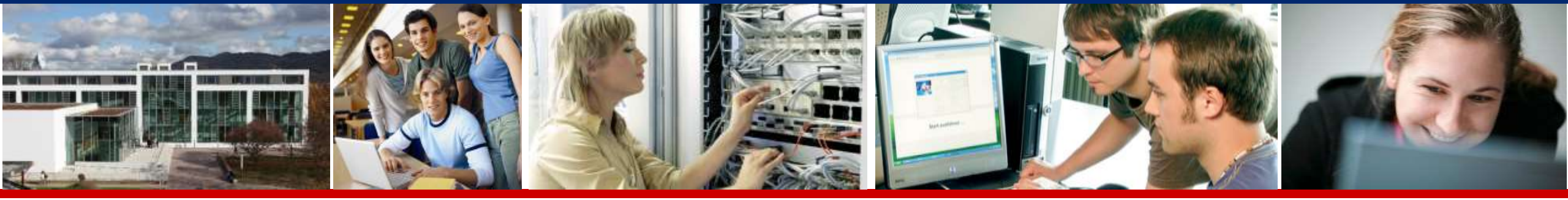


# Spark Machine Learning

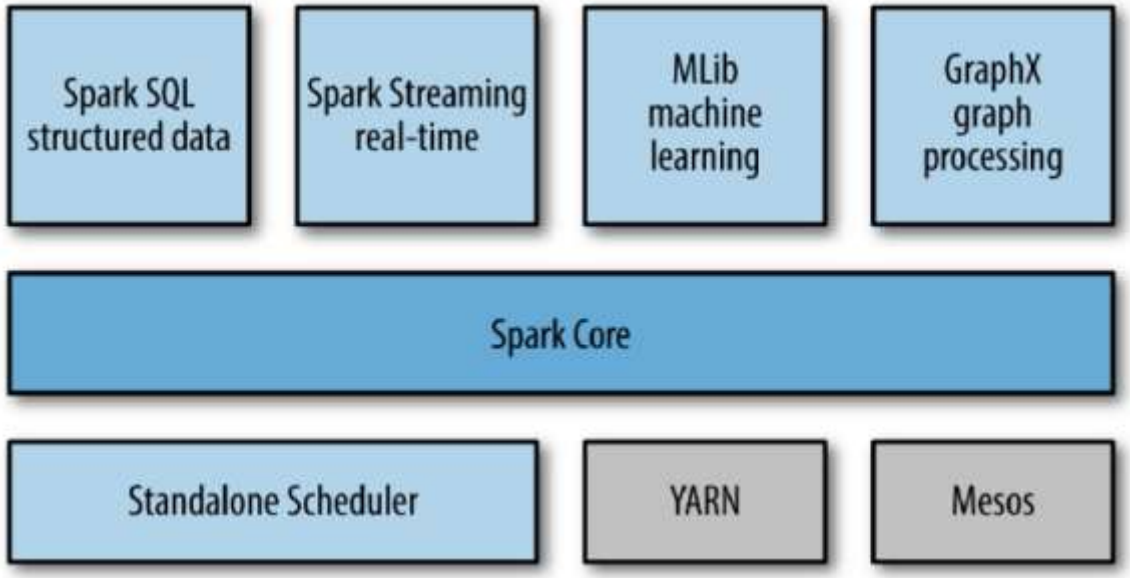
Prof. Dr. Stephan Trahasch
Hochschule Offenburg

# MLlib is Spark's machine learning (ML) library

Make practical machine learning scalable and easy.

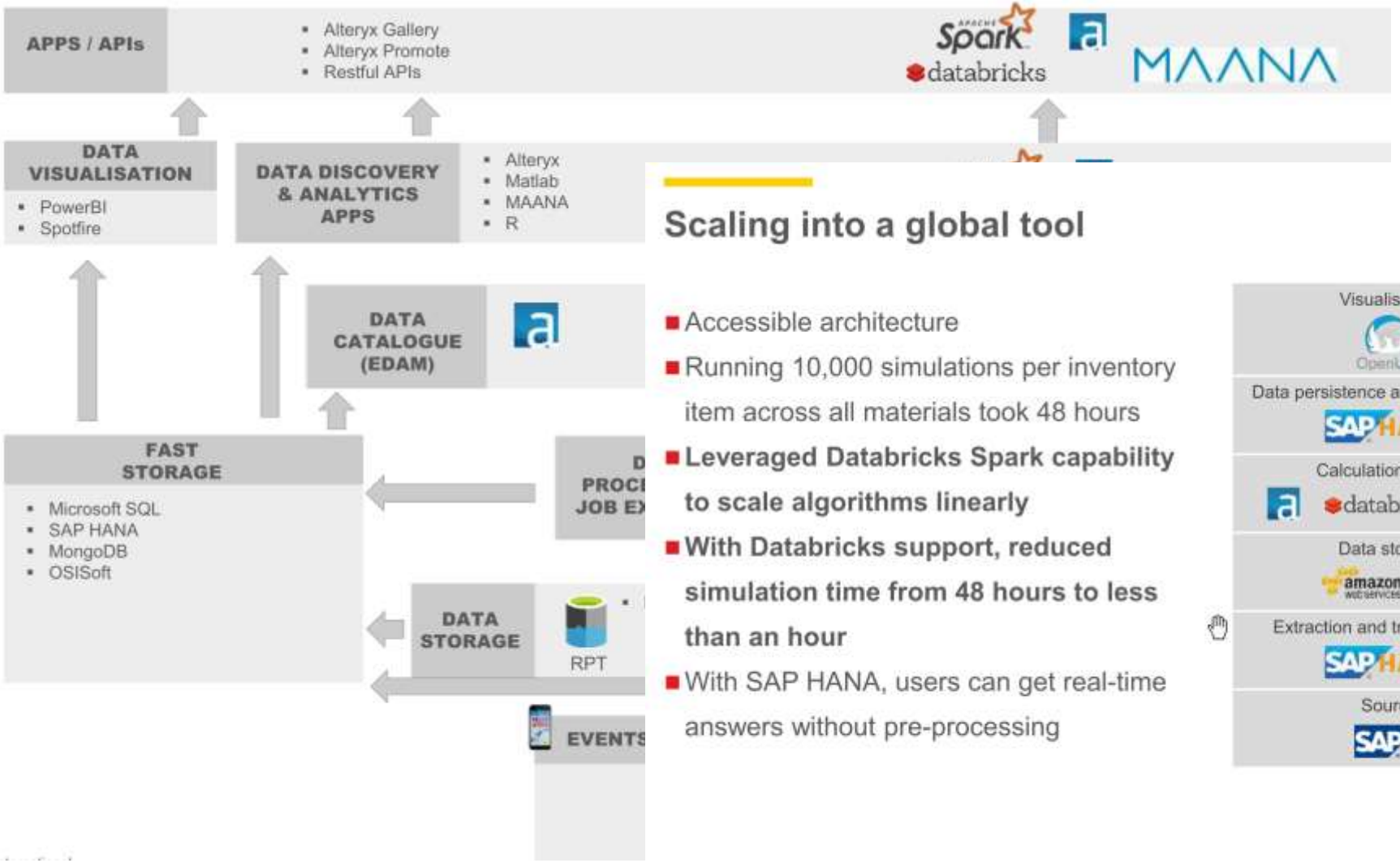
General purpose machine learning library optimized for big data

- Linearly scalable = 2x more machines, runtime theoretically cut in half
- Fault tolerant = resilient to the failure of nodes




- Covers the most common algorithms with distributed implementations
- Built around the concept of a Data Science Pipeline (scikit-learn)

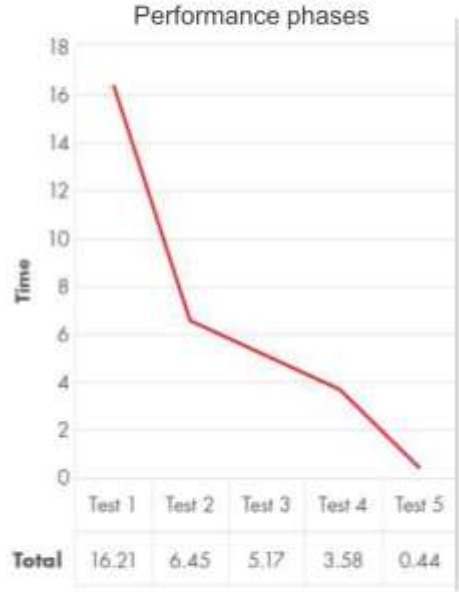
# Example: Shell




## Scaling into a global tool

- Accessible architecture
- Running 10,000 simulations per inventory item across all materials took 48 hours
- **Leveraged Databricks Spark capability to scale algorithms linearly**
- **With Databricks support, reduced simulation time from 48 hours to less than an hour**
- With SAP HANA, users can get real-time answers without pre-processing



# MLlib Tools

At a high level, it provides tools such as:

- ML Algorithms:
  learning algorithms such as classification, regression, clustering, and collaborative filtering
- Featurization:
  feature extraction, transformation, dimensionality reduction, and selection
- Pipelines:
  tools for constructing, evaluating, and tuning ML Pipelines
- Persistence:
  saving and load algorithms, models, and Pipelines
- Utilities:
  linear algebra, statistics, data handling, etc.

# DataFrame-based API since Spark 2.0

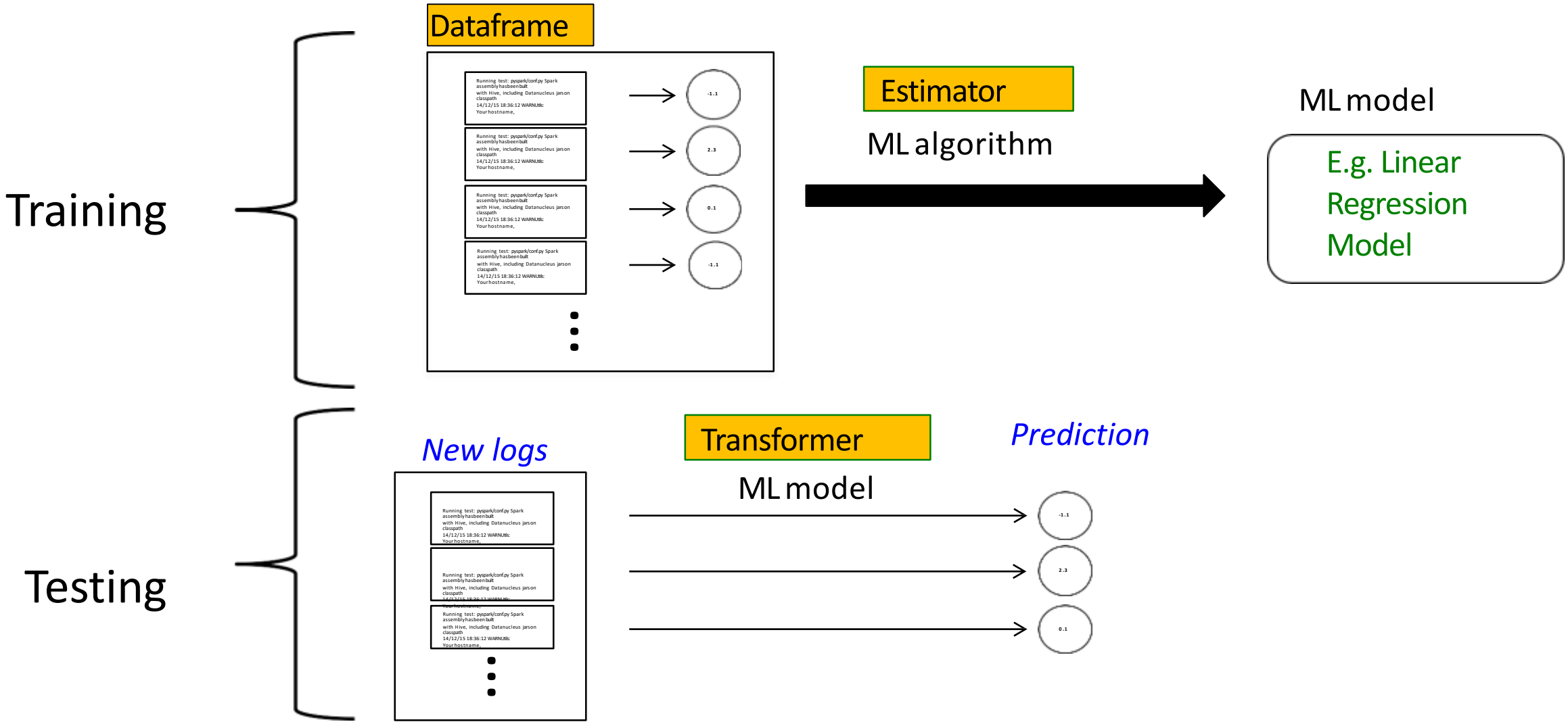
- Machine Learning API for Spark is the DataFrame-based API in the **spark.ml** package.
- As of Spark 2.0, the RDD-based APIs in spark.mllib is in maintenance mode.

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

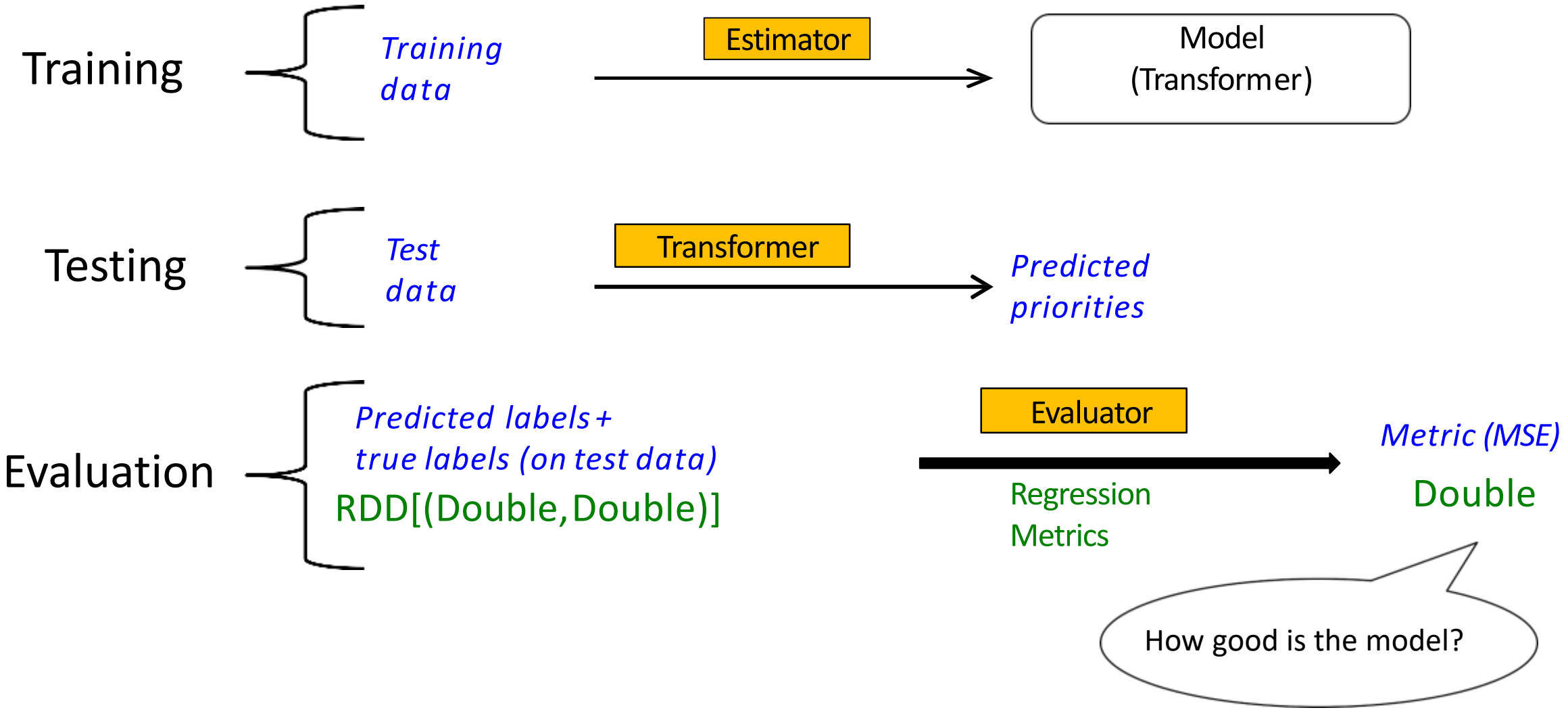
# ML algorithms include

- Statistics: summary statistics, hypothesis testing,...
- Regression: generalized linear regression, survival regression...
- Classification: logistic regression, naive Bayes,...
- Decision trees, random forests, and gradient-boosted trees
- Recommendation: alternating least squares (ALS)
- Clustering: K-means, Gaussian mixtures (GMMs),...
- Topic modeling: latent Dirichlet allocation (LDA)
- Frequent itemsets, association rules, and sequential pattern mining

# Pipeline: training + testing



# Pipeline


Training
Training data
Estimator
Model (Transformer)
Testing
Test data
Transformer
Predicted priorities
Evaluation
Predicted labels + true labels (on test data)
RDD[(Double, Double)]
Evaluator
Regression Metrics
Metric (MSE)
Double
How good is the model?

# ML workflow utilities include

- ML Pipeline construction
- Feature transformations: standardization, normalization, hashing,...
- Model evaluation and hyper-parameter tuning
- ML persistence: saving and loading models and Pipelines
- Distributed linear algebra: SVD, PCA,...

# Pipelines

- A Pipeline consists of a sequence of PipelineStages. Pipeline stages can be either estimators or transformers to be run in a specific order.



- These stages are run in order, and the input DataFrame is transformed as it passes through each stage.
- For Transformer stages, the transform() method is called on the DataFrame.
- For Estimator stages, the fit() method is called to produce a Transformer (which becomes part of the PipelineModel, or fitted Pipeline), and that Transformer's transform() method is called on the DataFrame.

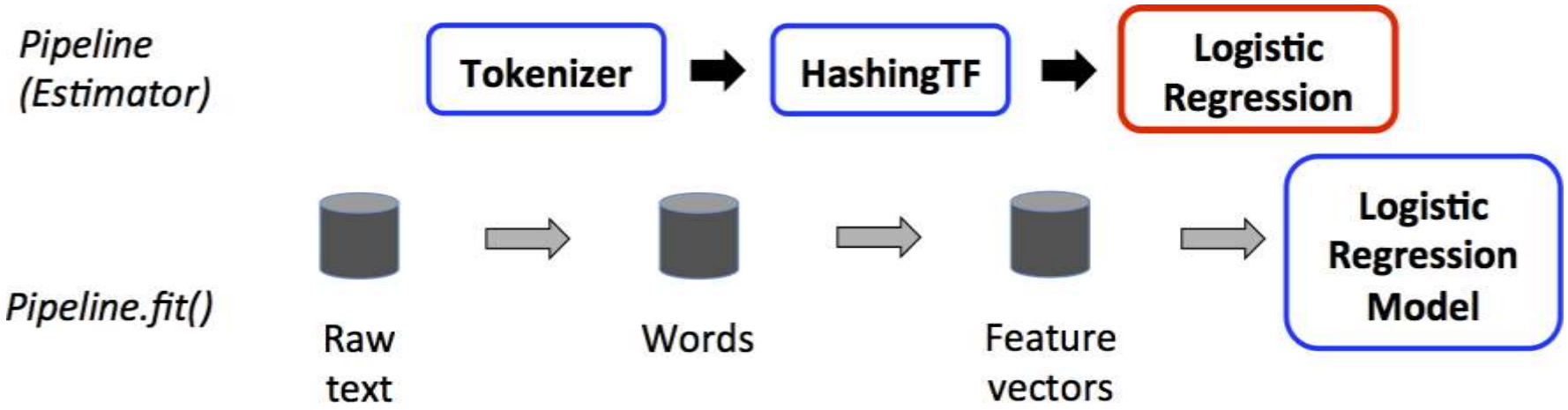
# Transformer

- A Transformer is a class which can transform one DataFrame into another DataFrame
- A Transformer implements method transform(), which converts one DataFrame into another
- Examples
  - HashingTF
  - Bucketizer
  - LogisticRegressionModel
  - PipelineModel
  - Binarizer

# Estimators

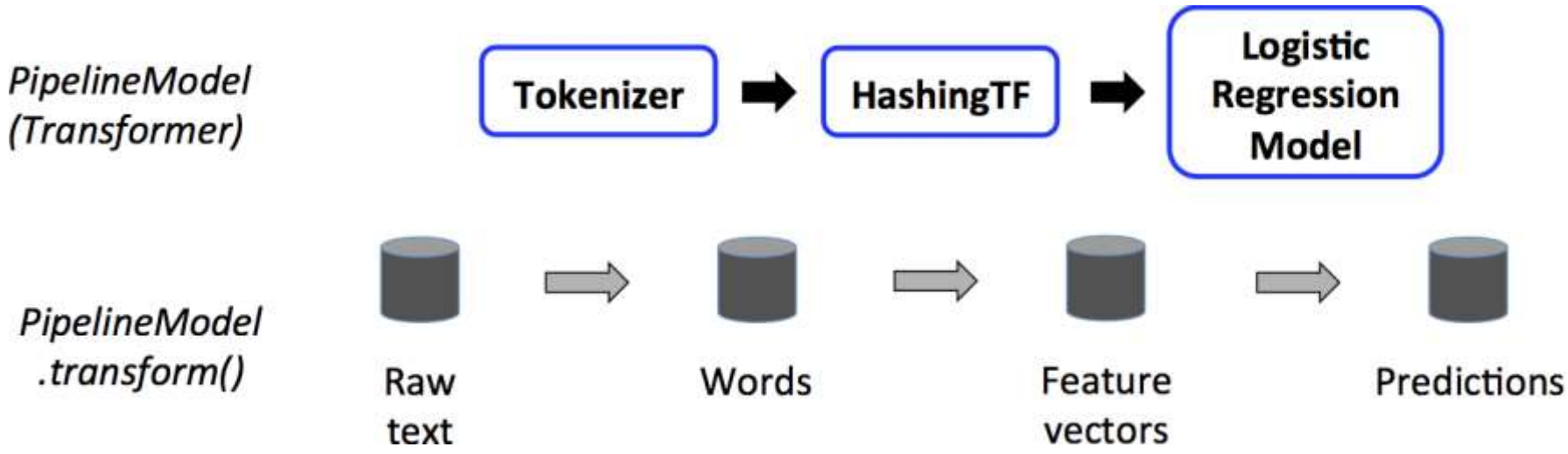
- An Estimator abstracts the concept of a learning algorithm or any algorithm that fits or trains on data.
- An Estimator implements a method fit(), which accepts a DataFrame and produces a Model, which is a Transformer.
- For example, a learning algorithm such as LogisticRegression is an Estimator, and calling fit() trains a LogisticRegressionModel, which is a Model and hence a Transformer.
- Examples
  - RandomForestClassifier
  - CrossValidator
  - IDF
  - Pipeline
  - StandardScaler

# Example Pipeline with fit()




- Pipeline with three stages. The first two (Tokenizer and HashingTF) are Transformers (blue), and the third (LogisticRegression) is an Estimator (red).
- The bottom row represents data flowing through the pipeline, where cylinders indicate DataFrames.

# Example PipelineModel with transform()




- Thus, after a Pipeline's fit() method runs, it produces a PipelineModel, which is a Transformer. This PipelineModel is used at test time.
- When the PipelineModel's transform() method is called on a test dataset, the data are passed through the fitted pipeline in order. Each stage's transform() method updates the dataset and passes it to the next stage.

# Pipeline and PipelineModel


# Example (Scala)

```
%spark2.spark  //Zeppelin Notebook
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.regression.{LinearRegression, LinearRegressionModel}

// Set Features
val features = new VectorAssembler().setInputCols(Array("x"))
       .setOutputCol("features")
val linreg = new LinearRegression().setLabelCol("y")
val pipeline = new Pipeline().setStages(Array(features, linreg))
val model = pipeline.fit(data)  // data is a dataframe (x,y)

val result = model.transform(data).select("x", "y", "prediction")

result.createOrReplaceTempView("linreg")
linreg.write.overwrite().save("hdfs:///tmp/linregmodel") //save model
val sameModel = LinearRegression.load("hdfs:///tmp/linregmodel")
```

# MLlib 2.X Model Serialization

## Data Science

- Develop Prototype Model using Scala/Python/R
- Persist model or Pipeline: model.save(“s3n://...”)
- Scala

```
val lrModel =
lrPipeline.fit(dataset)
// Save the Model
lrModel.write.save("/models/lr")
```

## Data Engineering

- Load Pipeline (Scala/Java) Model.load(“s3n://…”)
- Deploy in production
- Scala

# Persisted Models

```
lrModel.write.save("/models/lr")
```

```
%fs ls /models/lr
```

| path | name | size |
|---|---|---|
| dbfs:/models/lr/metadata/ | metadata/ | 0 |
| dbfs:/models/lr/stages/ | stages/ | 0 |

```
%fs ls /models/lr/metadata/
```

| path | name | size |
|---|---|---|
| dbfs:/models/lr/metadata/_SUCCESS | _SUCCESS | 0 |
| dbfs:/models/lr/metadata/part-00000 | part-00000 | 576 |

```
%fs head dbfs:/models/lr/metadata/part-00000
```


```
{"class":"org.apache.spark.ml.PipelineModel","timestamp":1494943789932,"sparkVersion":"2.1.0","uid":"pipeline_58b66ea70370","paramMap":{"stageUids":["strIdx_b8e750af3f46","oneHot_233aea43e7c9","strIdx_37dca6e88b05","oneHot_9798f7d45550","strIdx_84af213dc867","oneHot_11999cb41fce","strIdx_758c3e696aab","oneHot_ba258ff7c3b1","strIdx_85be2e39040d","oneHot_523a216d94a6","strIdx_c8f8890d9dd1","oneHot_e4acfe65e3e5","strIdx_39bf1a26e1df","oneHot_c0a595e1b547","strIdx_b135e86dc46a","oneHot_1865c0633b30","vecAssembler_29924bcb023e","strIdx_503a5a2c2185","logreg_b3003a237304"]}}
```


```
%fs ls /models/lr/stages/
```

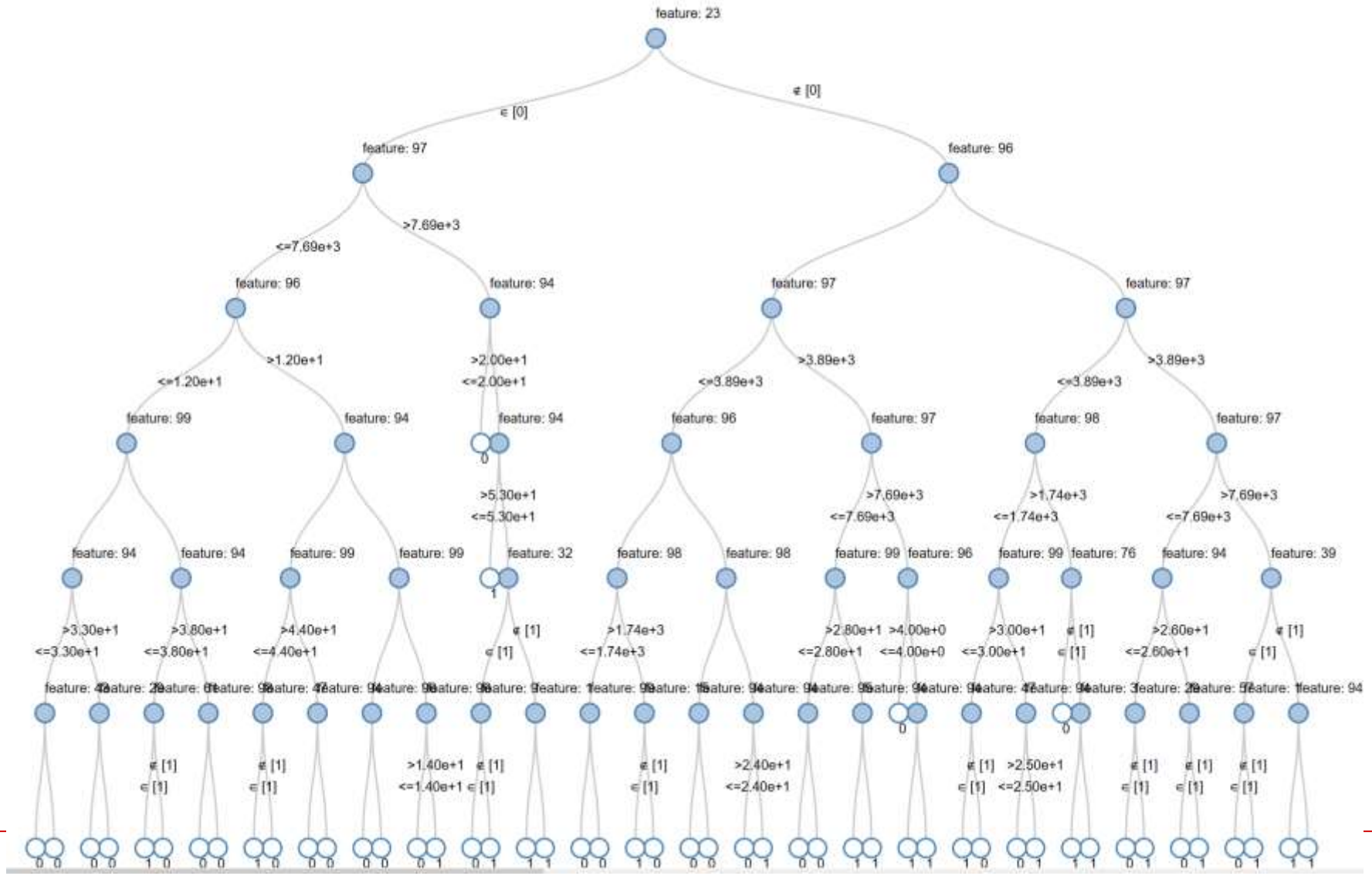
| path | name | size |
|---|---|---|
| dbfs:/models/lr/stages/00_strIdx_b8e750af3f46/ | 00_strIdx_b8e750af3f46/ | 0 |
| dbfs:/models/lr/stages/01_oneHot_233aea43e7c9/ | 01_oneHot_233aea43e7c9/ | 0 |
| dbfs:/models/lr/stages/02_strIdx_37dca6e88b05/ | 02_strIdx_37dca6e88b05/ | 0 |
| dbfs:/models/lr/stages/03_oneHot_9798f7d45550/ | 03_oneHot_9798f7d45550/ | 0 |
| dbfs:/models/lr/stages/04_strIdx_84af213dc867/ | 04_strIdx_84af213dc867/ | 0 |
| dbfs:/models/lr/stages/05_oneHot_11999cb41fce/ | 05_oneHot_11999cb41fce/ | 0 |
| dbfs:/models/lr/stages/06_strIdx_758c3e696aab/ | 06_strIdx_758c3e696aab/ | 0 |
| dbfs:/models/lr/stages/07_oneHot_ba258ff7c3b1/ | 07_oneHot_ba258ff7c3b1/ | 0 |

```
%fs head dbfs:/models/lr/stages/00_strIdx_b8e750af3f46/metadata/part-00000
```


```
{"class":"org.apache.spark.ml.feature.StringIndexerModel","timestamp":1494943790954,"sparkVersion":"2.1.0","uid":"strIdx_b8e750af3f46","paramMap":{"outputCol":"workclassIdx","inputCol":"workclass","handleInvalid":"error"}}
```

# Example: Decision Tree

```
display(dtModel.stages.last.asInstanceOf[DecisionTreeClassificationModel])
```



# Summary

- Further information at http://spark.apache.org/docs/latest/ml-guide.html

- Spark ML useful for Big Data Analytics
- Based on Dataframes and Pipelines
- Serialization of models helpful