Practical Machine Learning Pipelines with MLlib

Joseph K. Bradley

March 18, 2015

Spark Summit East 2015

About Spark MLlib

Started in UC Berkeley AMPLab

- Shipped with Spark 0.8

Currently (Spark 1.3)

- Contributions from 50+ orgs, 100+ individuals
- Good coverage of algorithms

classification	feature extraction, selection
regression	statistics
clustering	linear algebra
recommendation	frequent itemsets

MLlib's Mission

MLlib's mission is to make practical machine learning easy and scalable.

- Capable of learning from large-scale datasets
- Easy to build machine learning applications

How can we move beyond this list of algorithms and help users developer real ML workflows?

Outline

ML workflows

Pipelines

Roadmap

Outline

ML workflows

Pipelines

Roadmap

Example: Text Classification

Goal: Given a text document, predict its topic.

Dataset: "20 Newsgroups"
From UCI KDD Archive

Features

```
Subject: Re: Lexan Polish?
Suggest McQuires #1 plastic
polish.  It will help somewhat
but nothing will remove deep
scratches without making it
worse than it already is.
McQuires will do something...
```

text, image, vector, ...

Label

1: about science
0: not about science

CTR, inches of rainfall, ...

Training & Testing

Training

Given labeled data:
RDD of (features, label)

```
Subject: Re: Lexan Polish?
Suggest McQuires #1 plastic
polish.  It will help...
```
Label 0

```
Subject: RIPEM FAQ
RIPEM is a program which
performs Privacy Enhanced...
```
Label 1

...

Learn a model.

Testing/Production

Given new unlabeled data:
RDD of features

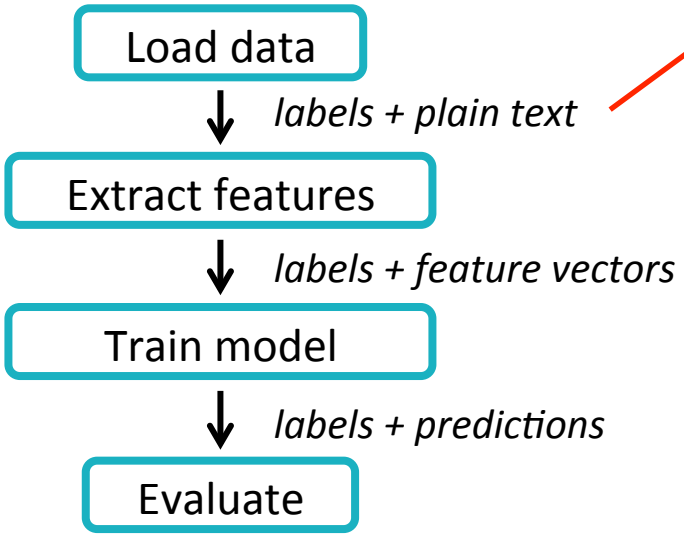
```
Subject: Apollo Training
The Apollo astronauts also
trained at (in) Meteor...
```
→ Label 1

```
Subject: A demo of Nonsense
How can you lie about
something that no one...
```
→ Label 0

Use model to make predictions.

Example ML Workflow

Training



Pain point

Create many RDDs

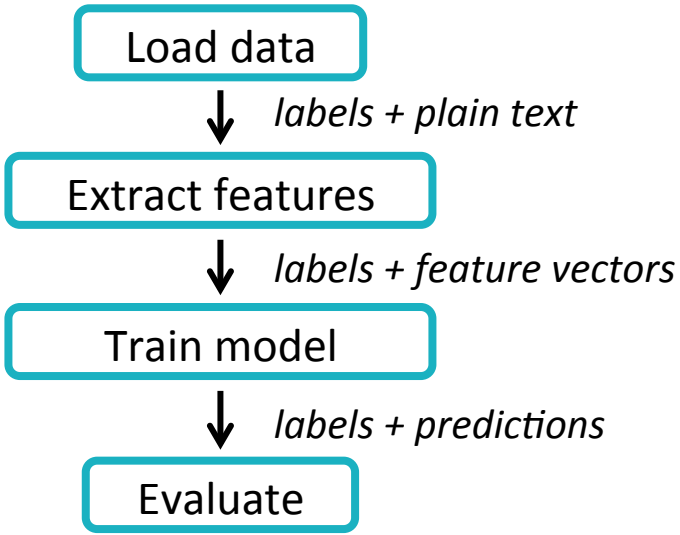
```
val labels: RDD[Double] =
  data.map(_.label)

val features: RDD[Vector]
val predictions: RDD[Double]
```

Explicitly unzip & zip RDDs

```
labels.zip(predictions).map {
  if (_._1 == _._2) ...
}
```

Example ML Workflow

Training



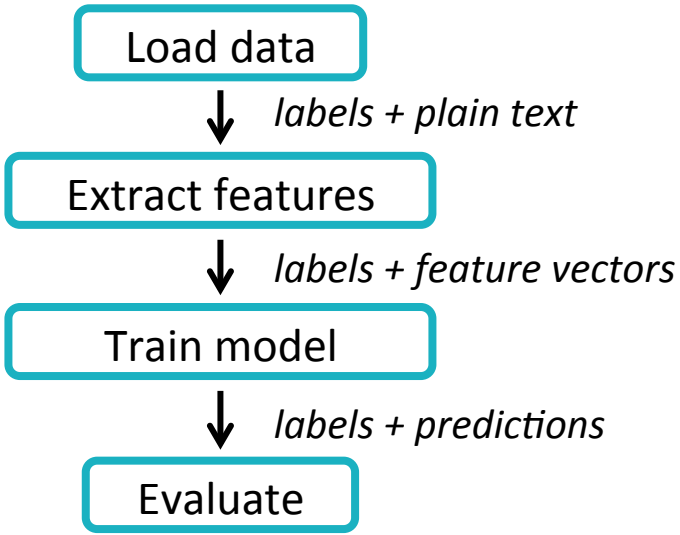
Pain point

Write as a script

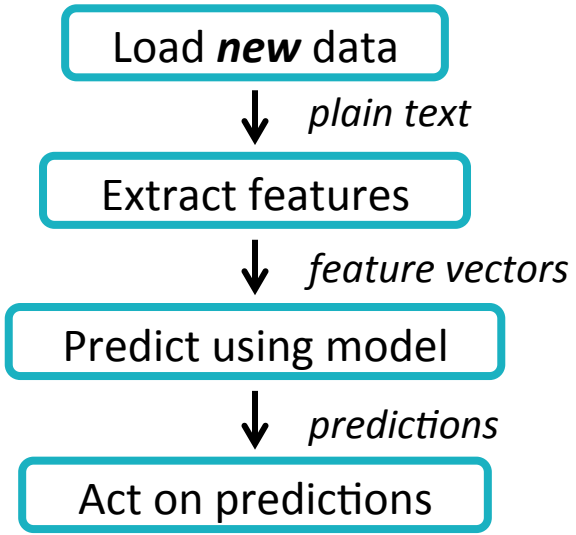
- Not modular
- Difficult to re-use workflow

Example ML Workflow

Training



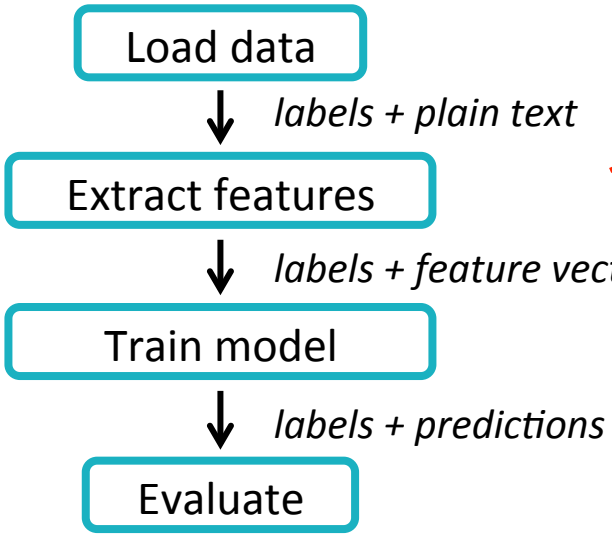
Testing/Production



Almost identical workflow

Example ML Workflow

Training



Pain point

Parameter tuning

- Key part of ML
- Involves training many models
  - For different splits of the data
  - For different sets of parameters

Pain Points

Create & handle many RDDs and data types
Write as a script
Tune parameters

Enter...

Pipelines!

in Spark 1.2 & 1.3

Outline

ML workflows

Pipelines

Roadmap

Key Concepts

DataFrame: The ML Dataset

Abstractions: Transformers, Estimators, & Evaluators

Parameters: API & tuning

DataFrame: The ML Dataset

DataFrame: RDD + schema + DSL

Named columns with types

```
label: Double
text: String
words: Seq[String]
features: Vector
prediction: Double
```

label	text	words	features
0	This is ...	[“This”, “is”, …]	[0.5, 1.2, …]
0	When we ...	[“When”, ...]	[1.9, -0.8, …]
1	Knuth was ...	[“Knuth”, …]	[0.0, 8.7, …]
0	Or you ...	[“Or”, “you”, …]	[0.1, -0.6, …]

DataFrame: The ML Dataset

DataFrame: RDD + schema + DSL

Named columns with types

Domain-Specific Language

```
# Select science articles
sciDocs =
  data.filter("label" == 1)

# Scale labels
data("label") * 0.5
```

DataFrame: The ML Dataset

DataFrame: RDD + schema + DSL

BIG data

Named columns with types

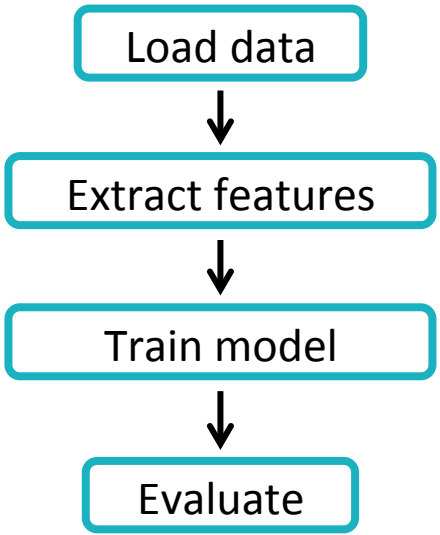
Domain-Specific Language

- Shipped with Spark 1.3
- APIs for Python, Java & Scala (+R in dev)
- Integration with Spark SQL
  - Data import/export
  - Internal optimizations

~~Pain point: Create & handle many RDDs and data types~~

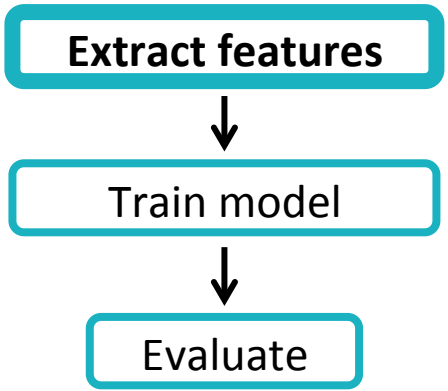
Abstractions

Training

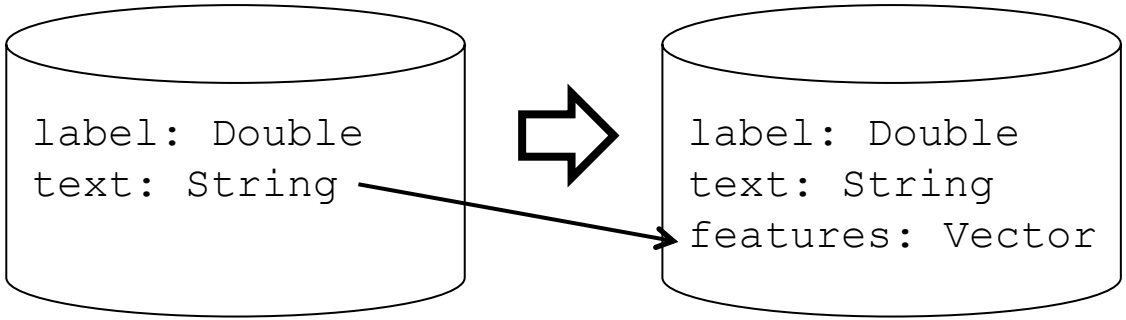


Abstraction: Transformer

Training



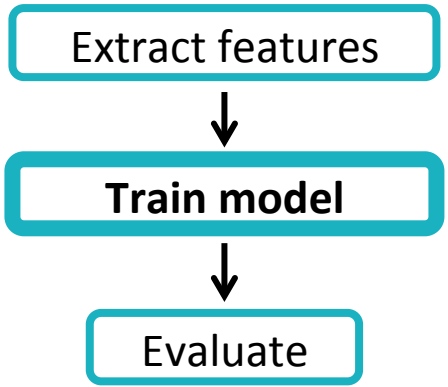

```
def transform(DataFrame): DataFrame
```

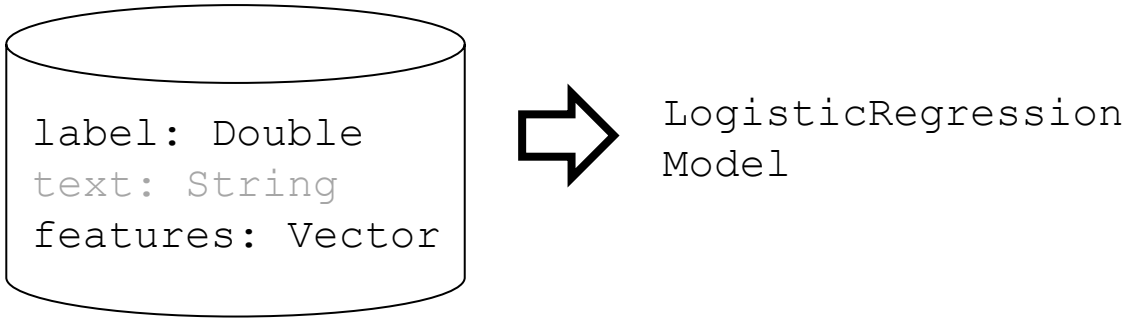


Abstraction: Estimator

Training

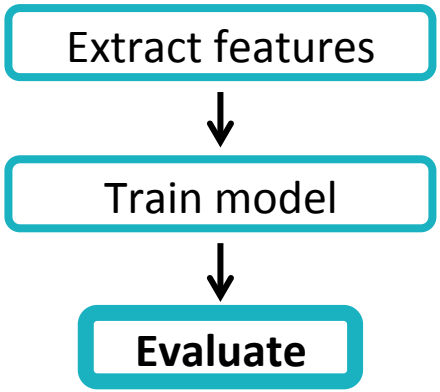
```
def fit(DataFrame): Model
```



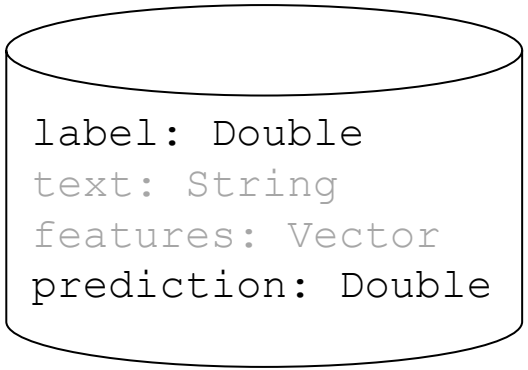


Abstraction: Evaluator

Training




```
def evaluate(DataFrame): Double
```

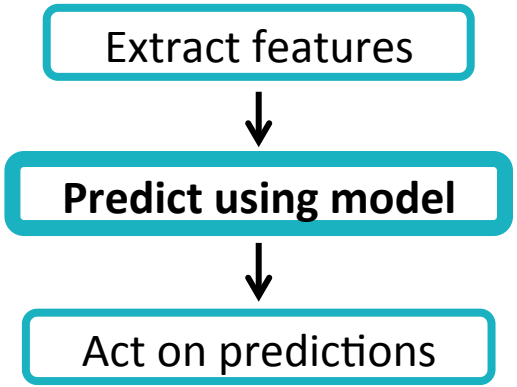


Metric:

```
accuracy
AUC
MSE
...
```

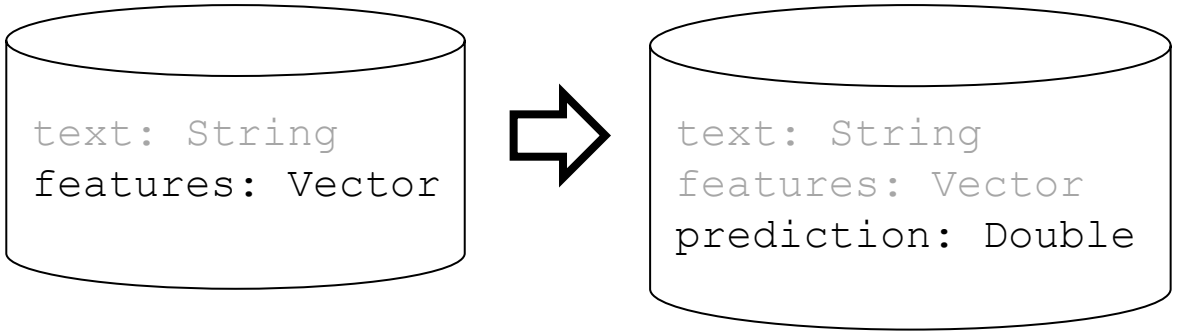
Abstraction: Model

Testing/Production



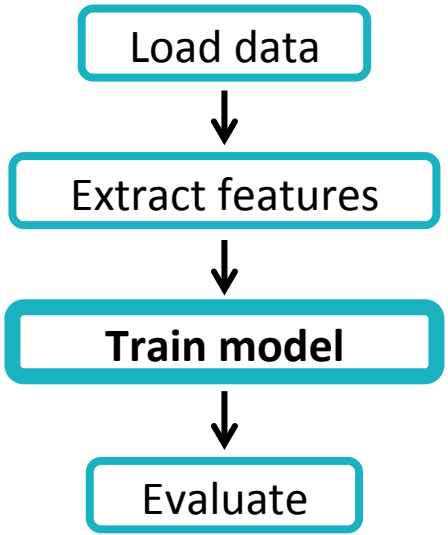
Model is a type of Transformer

```
def transform(DataFrame): DataFrame
```

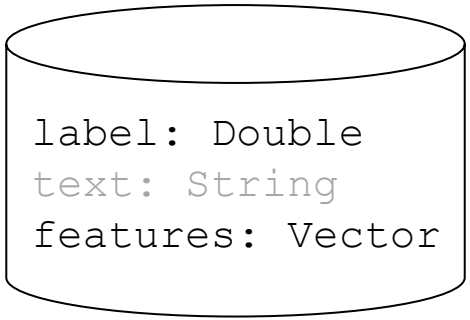


(Recall) Abstraction: Estimator

Training

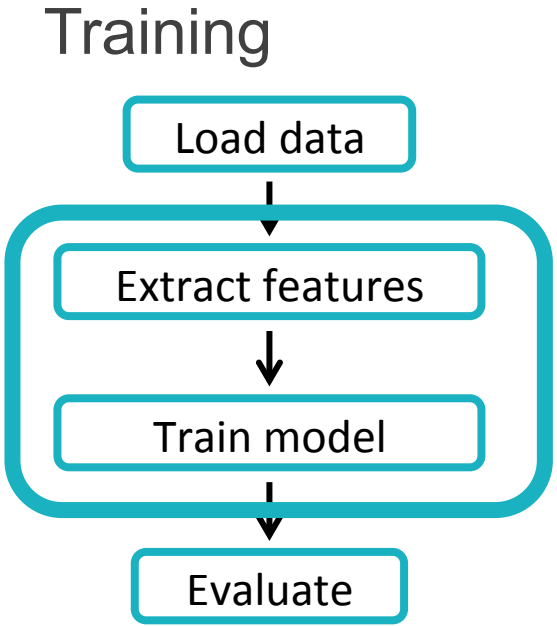



```
def fit(DataFrame): Model
```



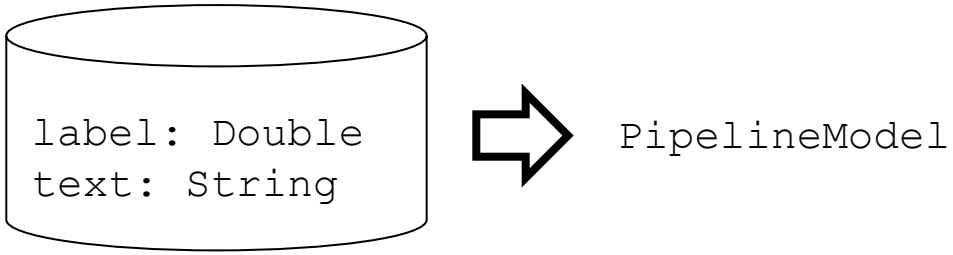
```
LogisticRegression
Model
```

Abstraction: Pipeline



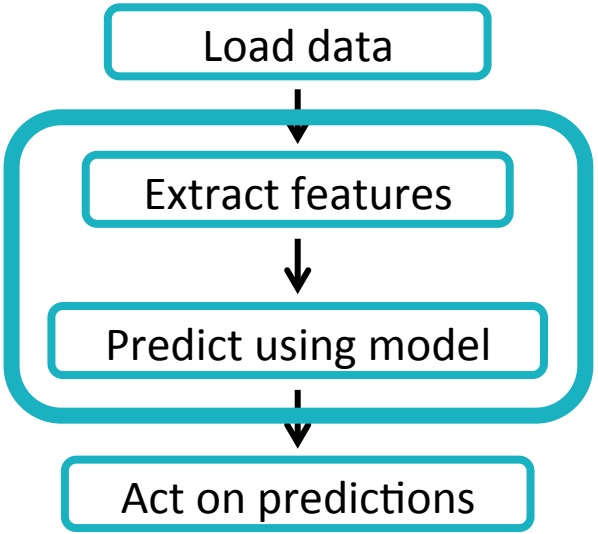
Pipeline is a type of Estimator

```
def fit(DataFrame): Model
```



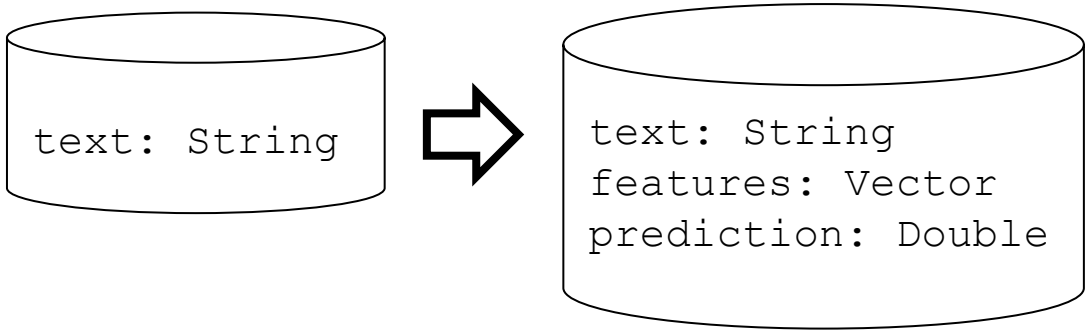
Abstraction: PipelineModel

Testing/Production



PipelineModel is a type of Transformer

```
def transform(DataFrame): DataFrame
```



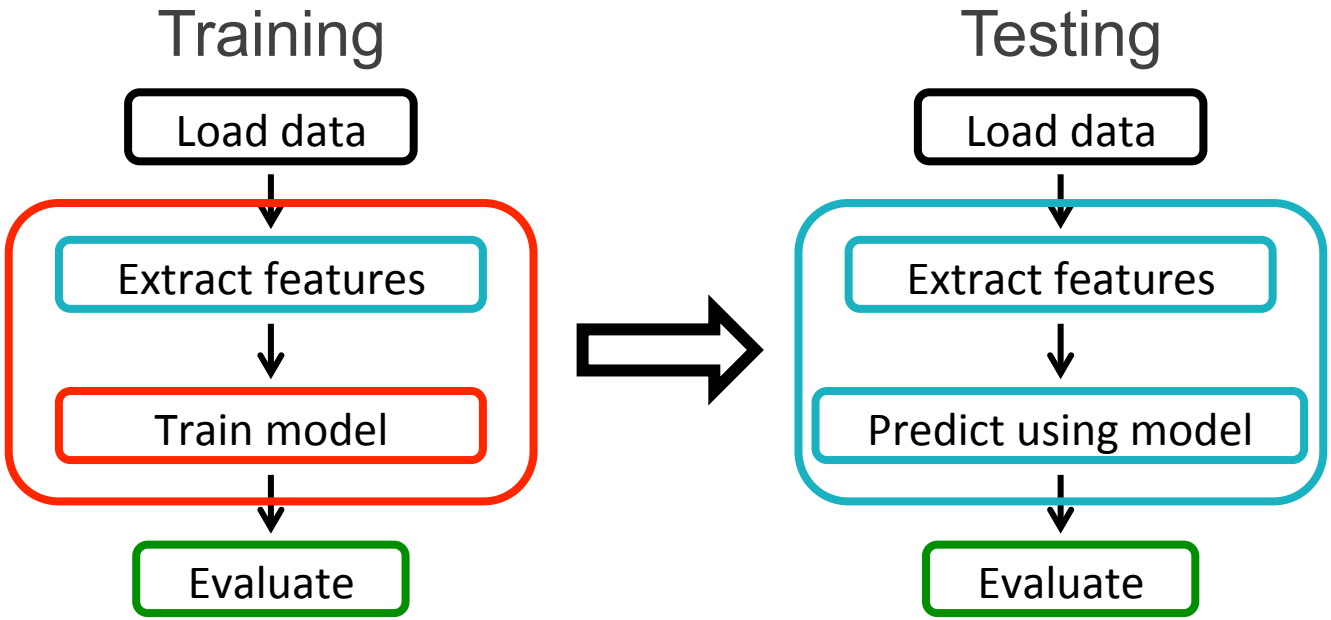
Abstractions: Summary

DataFrame

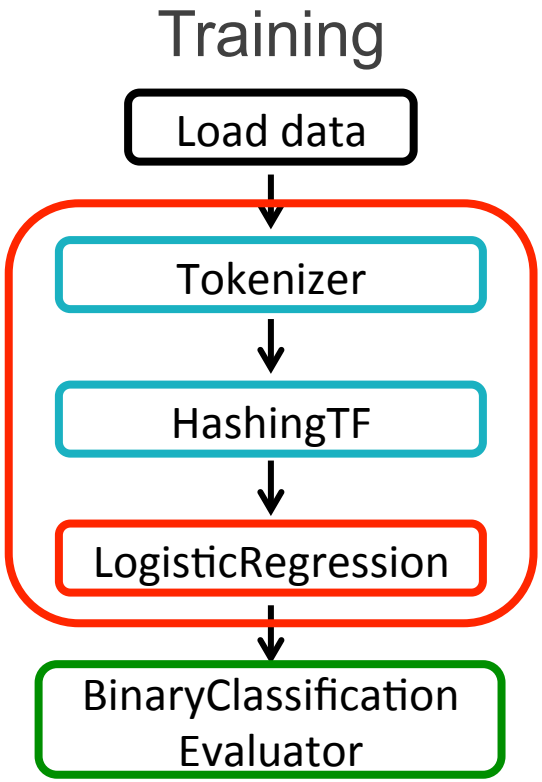
Transformer

Estimator

Evaluator



Demo



DataFrame

Transformer

Transformer

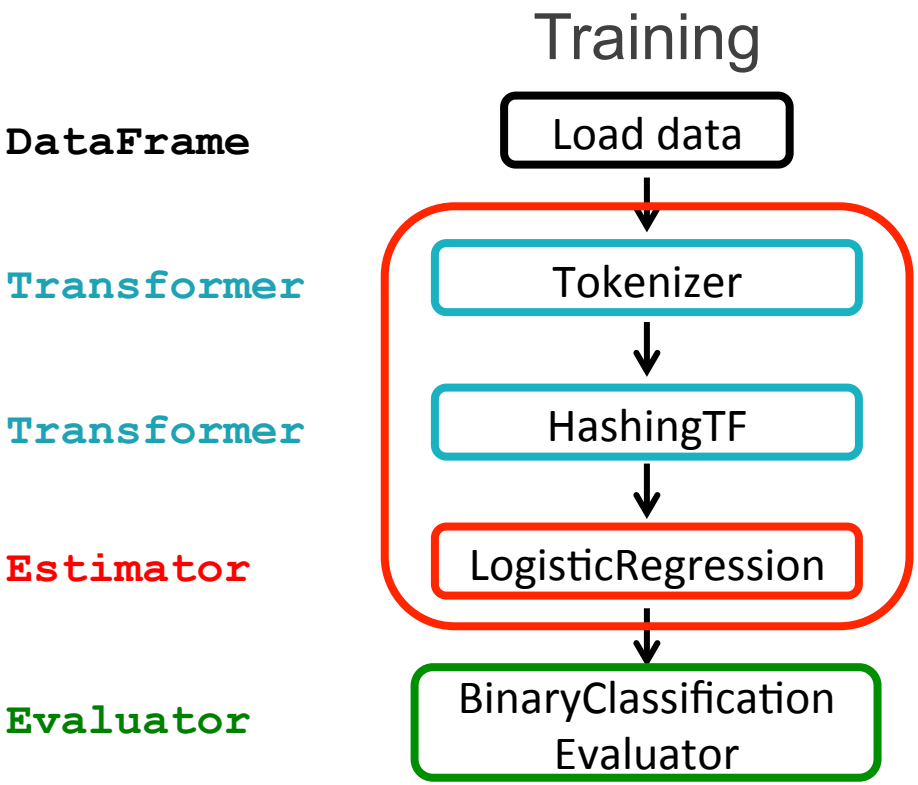
Estimator

Evaluator

Current data schema

```
label: Double
text: String

words: Seq[String]

features: Vector

prediction: Double
```

Demo



Pain point: Write as a script

Parameters

Standard API

- Typed
- Defaults
- Built-in doc
- Autocomplete

```
> hashingTF.numFeatures

    org.apache.spark.ml.param.IntParam =
     numFeatures: number of features
     (default: 262144)

> hashingTF.setNumFeatures(1000)

> hashingTF.getNumFeatures
```

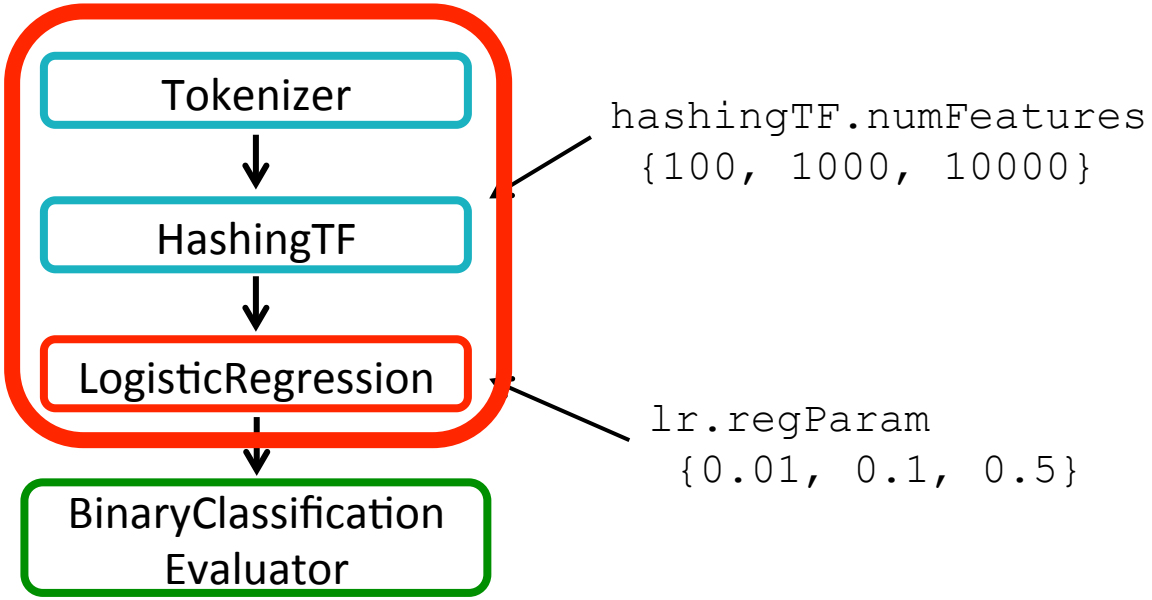
Parameter Tuning

Given:

- Estimator
- Parameter grid
- Evaluator

Find best parameters

`CrossValidator`



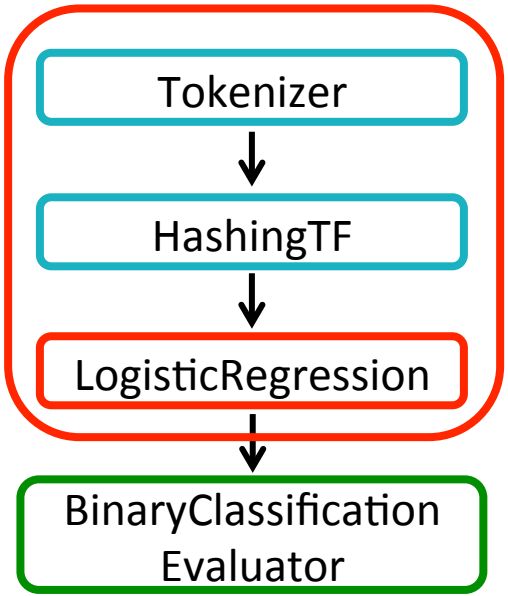
Parameter Tuning

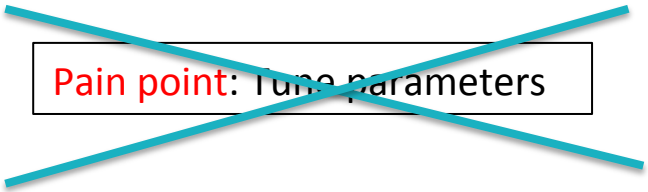
Given:

- Estimator
- Parameter grid
- Evaluator

Find best parameters

`CrossValidator`





Pipelines: Recap

DataFrame → *Create & handle many RDDs and data types*
Abstractions → *Write as a script*
Parameter API → *Tune parameters*

Also
- Python, Scala, Java APIs
- Schema validation
- User-Defined Types*
- Feature metadata*
- Multi-model training optimizations*

Inspirations

scikit-learn
+ Spark DataFrame, Param API

MLBase (Berkeley AMPLab)
Ongoing collaborations

** Groundwork done; full support WIP.*

Outline

ML workflows

Pipelines

Roadmap

Roadmap

`spark.mllib`: Primary ML package

`spark.ml`: High-level Pipelines API for algorithms in `spark.mllib` (*experimental in Spark 1.2-1.3*)

Near future

- Feature attributes
- Feature transformers
- More algorithms under Pipeline API

Farther ahead

- Ideas from AMPLab MLBase (auto-tuning models)
- SparkR integration

Outline

- **ML workflows**
- **Pipelines**
  - **DataFrame**
  - **Abstractions**
  - **Parameter tuning**
- **Roadmap**

Spark documentation
http://spark.apache.org/

Pipelines blog post
https://databricks.com/blog/2015/01/07

Thank you!

databricks™