Introduction to MLlib

¡Hola!

Sergio Ramírez

sramirez@decsai.ugr.es

Ph. D candidate from University of Granada, and member of the *SCI²S* research group
and that's what I do:
http://sci2s.ugr.es/BigData
https://scholar.google.es/citations?user=cEkfaW4AAAAJ
https://github.com/sramirez

Outline

1. **Acerca de MLLIB**
2. **Breve historia**
3. **Objetivos**
4. **ML Workflows**
5. **Importación/Exportación de datos**
6. **ML Pipelines**
   1. DataFrames
   2. Entrenamiento multi-modelo
   3. Ajuste de parámetros
7. **Novedades**

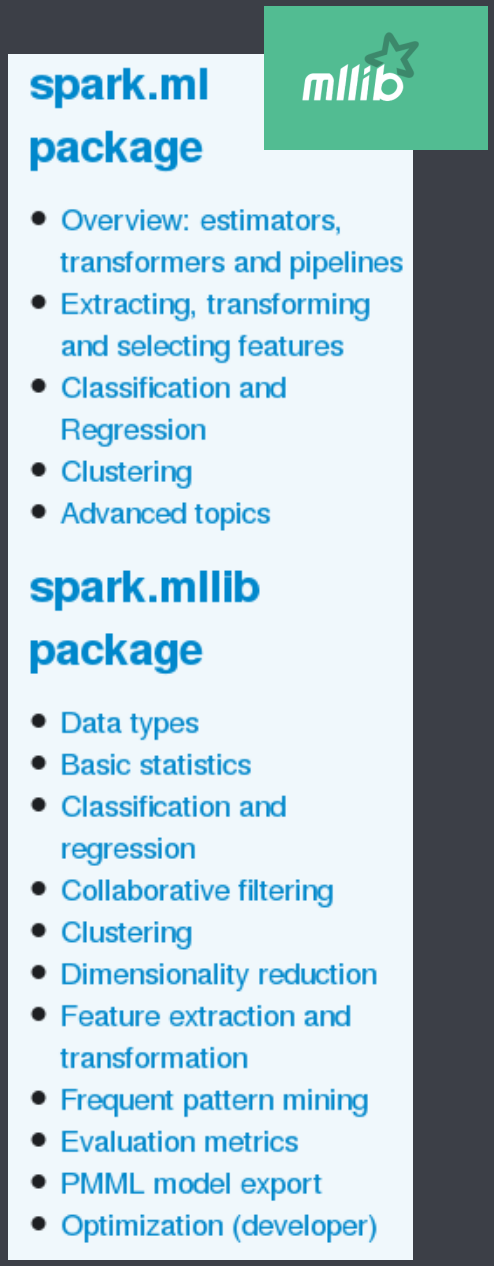
Acerca de MLLIB

Proyecto iniciado en UC Berkeley AMPLab

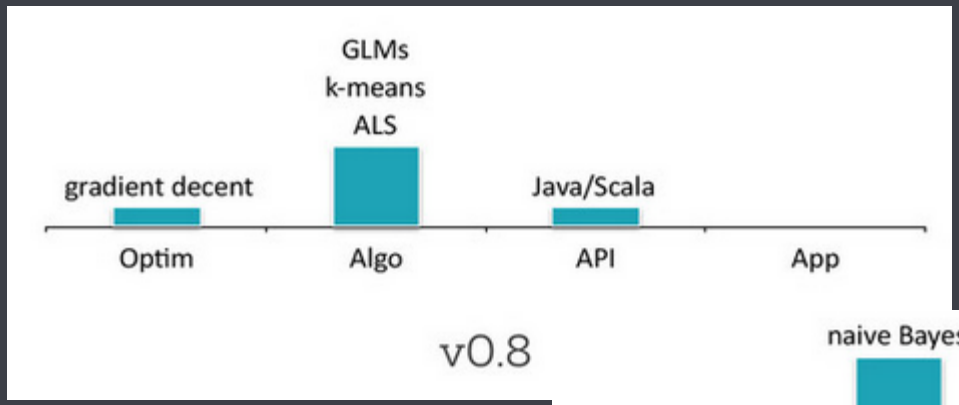
- Primera versión con Spark 0.8

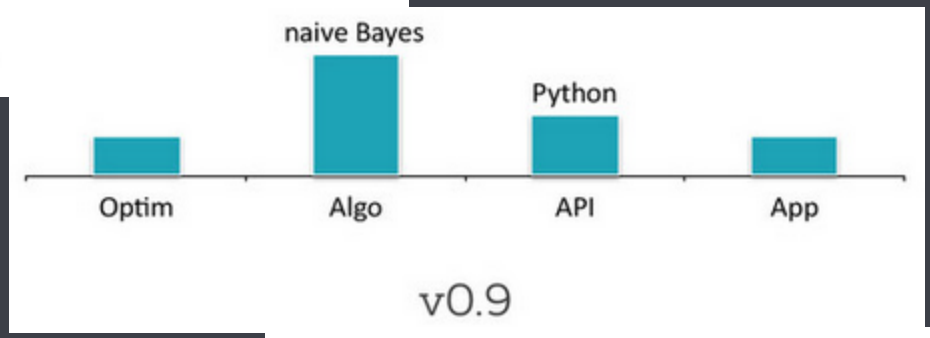
Versión actual (Spark 1.6)

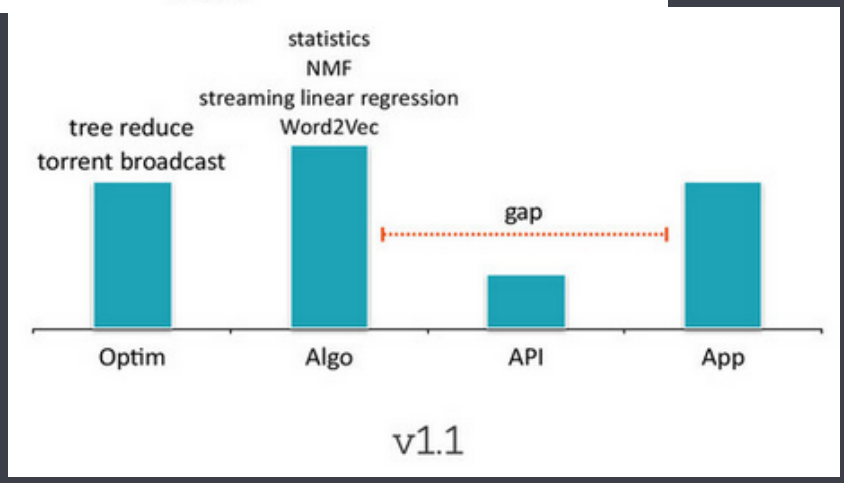
- Contribuciones de 50+ organizaciones y 110+ particulares.
- Buena cobertura de algoritmos



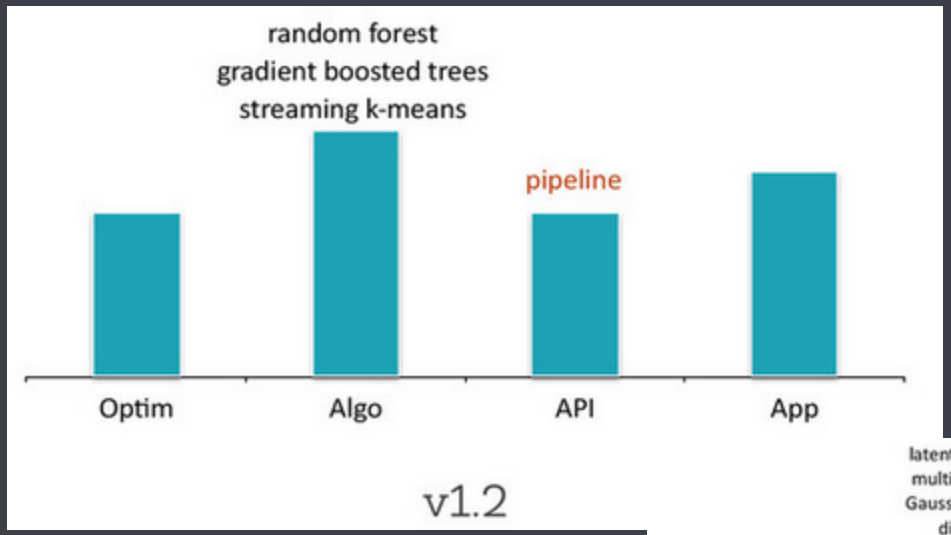
Breve historia de MLLIB

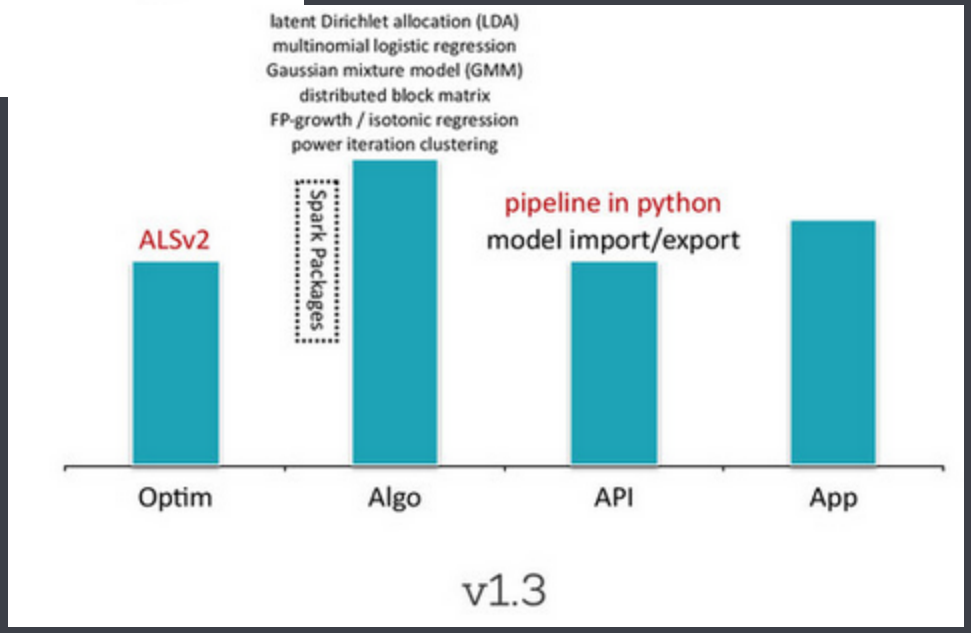






Breve historia de MLLIB



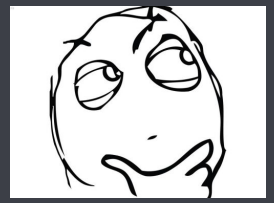


Objetivo de MLLIB

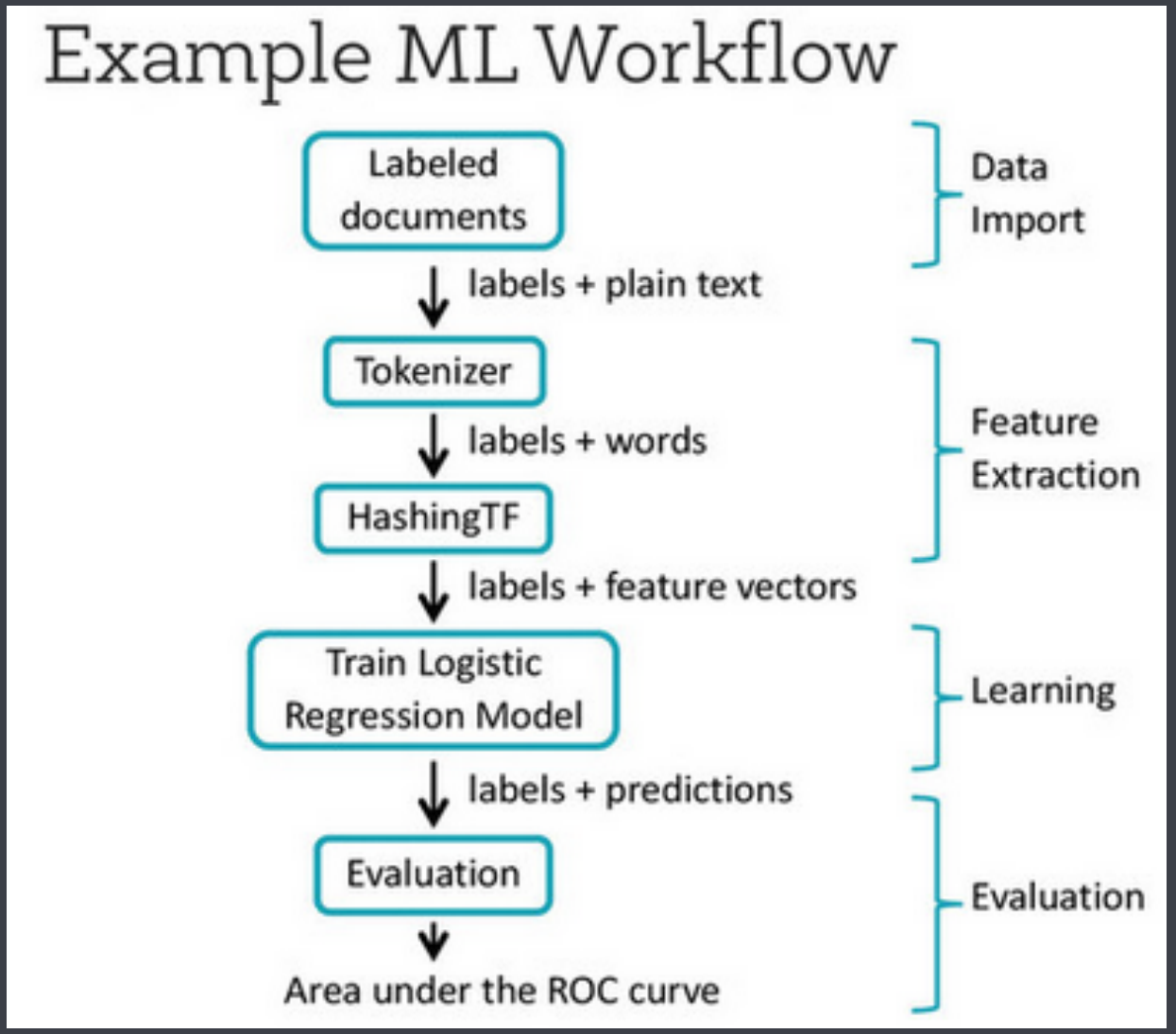
El objetivo principal de MLLIB es aportar una herramienta para el aprendizaje automático, que sea fácil y escalable.

- Fácil construcción de aplicaciónes para machine learning.
- Capaz de manejar conjuntos de datos de gran tamaño (millones de instancias y atributos).

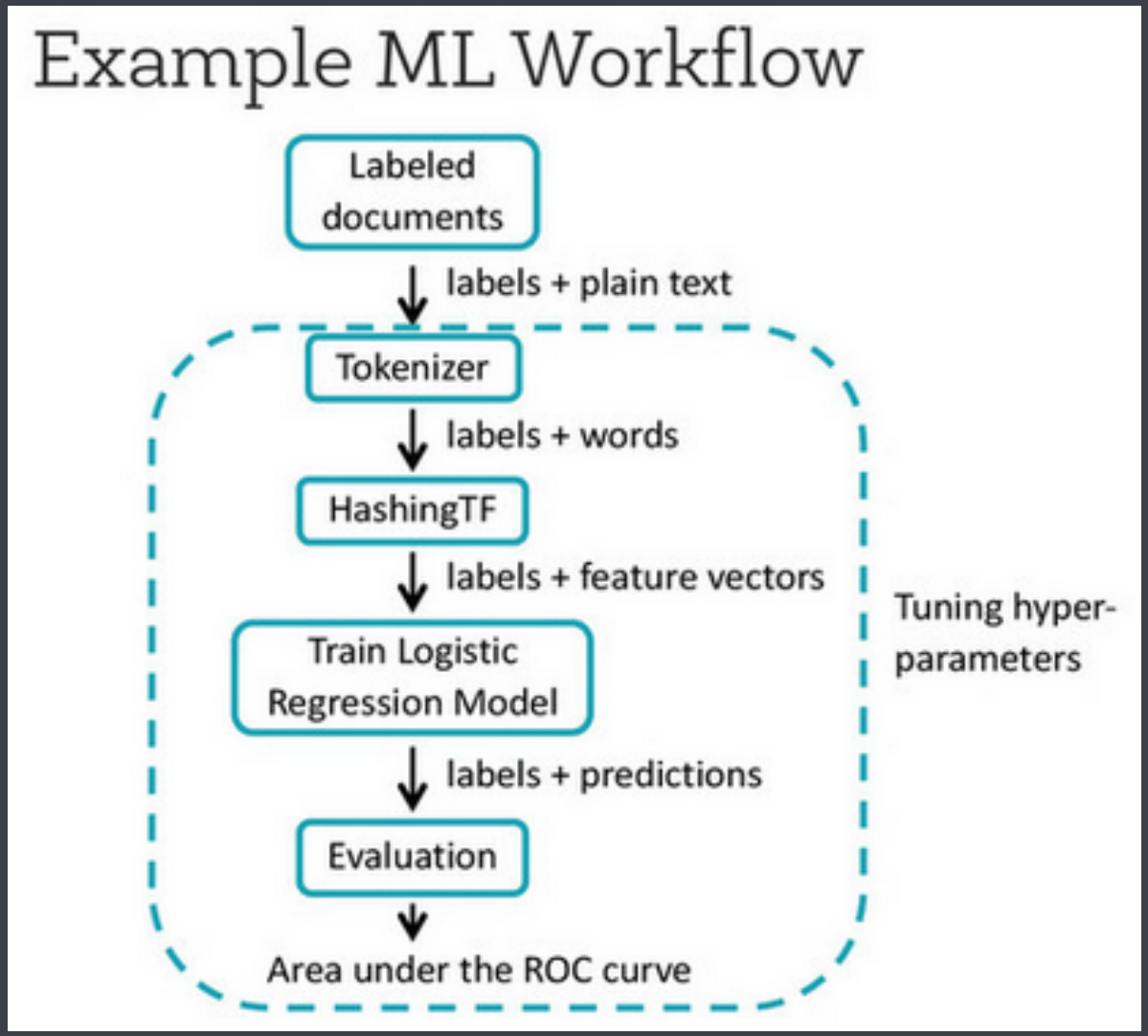
¿Pero cómo ir más allá de la simple aportación de algoritmos escalables?
¿Cómo ayudar al usuario a desarrollar auténticos procesos de ML?



ML Workflow


Example ML Workflow
Labeled documents
labels + plain text
Tokenizer
labels + words
HashingTF
labels + feature vectors
Train Logistic Regression Model
labels + predictions
Evaluation
Area under the ROC curve
Data Import
Feature Extraction
Learning
Evaluation

ML Workflow


Example ML Workflow
Labeled documents
labels + plain text
Tokenizer
labels + words
HashingTF
labels + feature vectors
Train Logistic Regression Model
labels + predictions
Evaluation
Area under the ROC curve
Tuning hyper-parameters

ML Pipelines

Importación/Exportación de datos:

- **LIBSVM format**: 0 128:51.0 130:253.0 155:48.0
  MlUtils.saveAsLibSVMFile / MLUtils.loadLibSVMFile
- **Mllib format**: (0,[127,129,154], [51.0, 253.0, 48.0])
  MlUtils.loadLabeledPoints
- **Java serialization / Python pickling**:
  object.writeObject()

Es necesario un formato estándar:

- Compatible con **dataset de tipos específicos** en lugar de RDD[LabeledPoint]. E.g.: RDD[(Int, Vector, Vector)].
- Que pueda almacenar **información relacionada con el proceso ML**.
- Soporte para **almacenamiento por columnas y comprensión**.
- Soporte para **cargar y guardar en varios sistemas**.

ML Pipelines

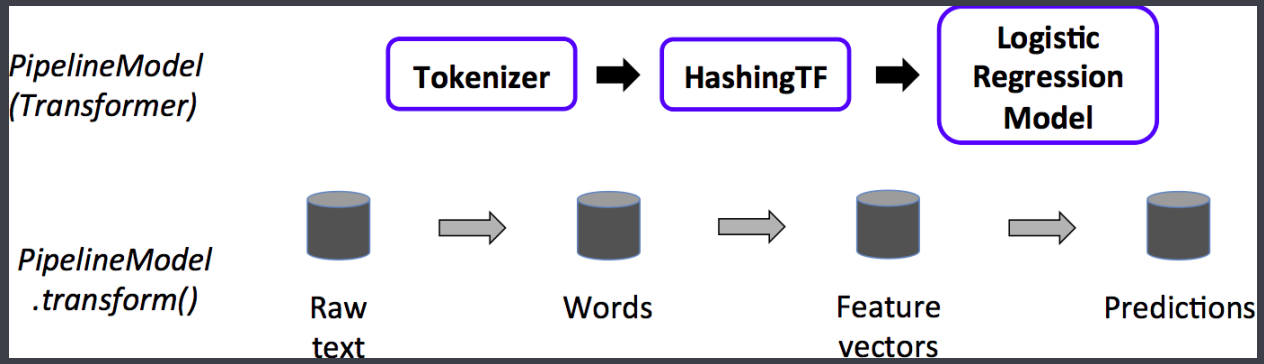
Difícil de definir un workflow:

- Escribir tu propio código para unir componentes para el entrenamiento y test **(¡bienvenido a mi mundo!).**

No soporte para ajuste de hiper-parámetros:

- Difícil de referenciar los hiper-parámetros fuera de los algoritmos.
- Escribir tu propio código iterativo para encontrar el mejor modelo.

¡ML PIPELINES!



DataFrames

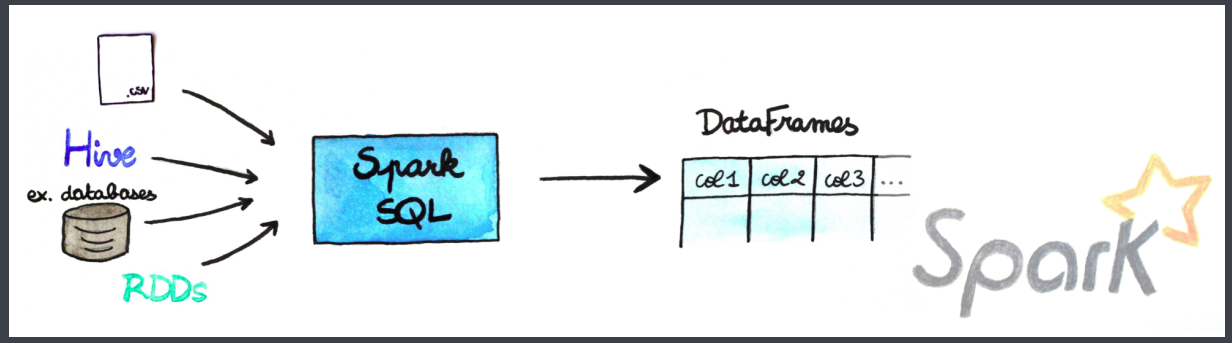
Facilidad de que algo falle en un ML workflow. Es importante chequear el proceso:

- Sin embargo, no es sencillo inspeccionar resultado mediante campos extra:

```
words = tokenizer.transform(docs.map(lambda x: (x.id, x.text)))
features = words.map(lambda x: (x[0], tf.transform(x[1])))
```

Solución: DataFrames (mllib 1.3)

- Columnas con nombre
- Operaciones declarativas con optimización del plan de ejecución.
- Soporte para datos **densos/dispersos** via UDTs
- **Almacenamiento por columnas** para varios tipos (Parquet).



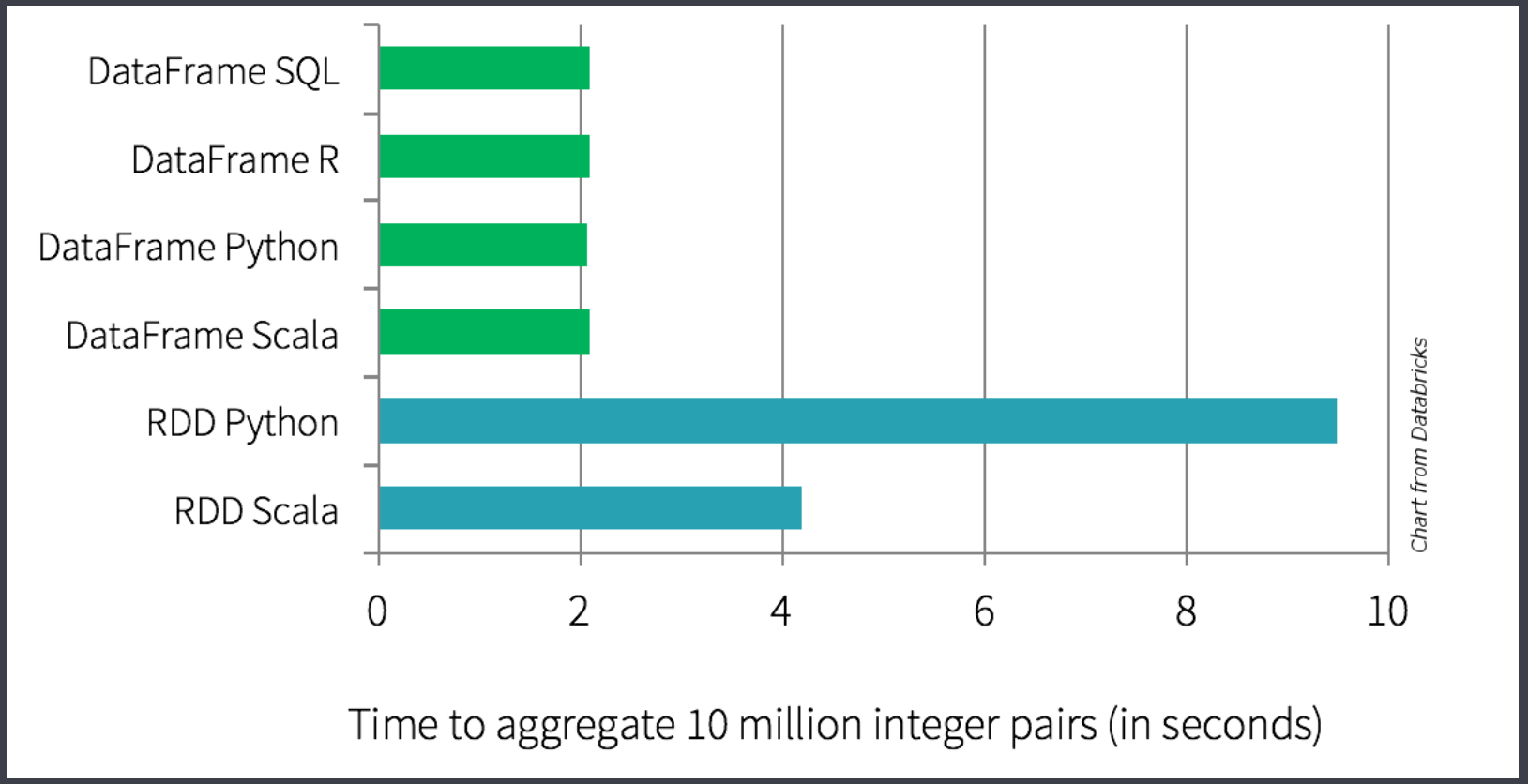
RDD API vs. DataFrames API

```
data.map(lambda x: (x.dept, [x.age, 1])) \
 .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
 .map(lambda x: [x[0], x[1][0] / x[1][1]])
```

vs.

```
data.groupBy("dept").avg("age")
```

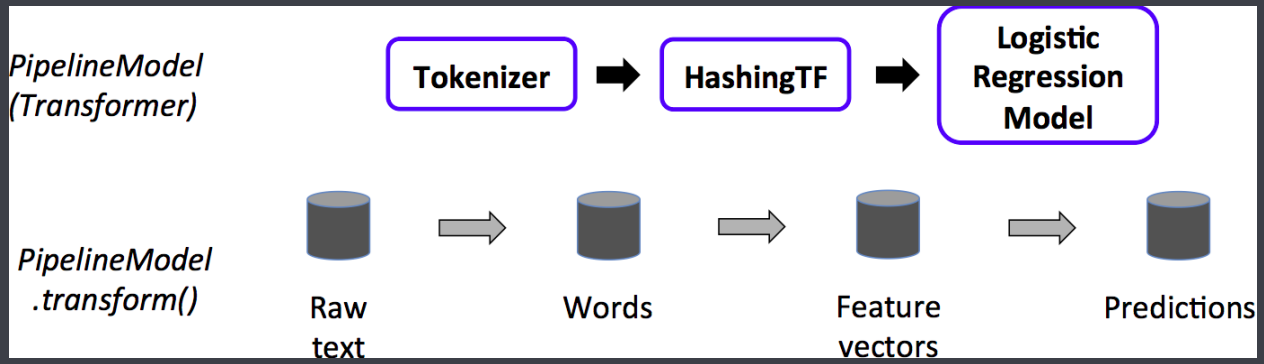
DataFrames


DataFrame SQL
DataFrame R
DataFrame Python
DataFrame Scala
RDD Python
RDD Scala
0
2
4
6
8
10
Chart from Databricks
Time to aggregate 10 million integer pairs (in seconds)

ML Pipelines

Componentes:

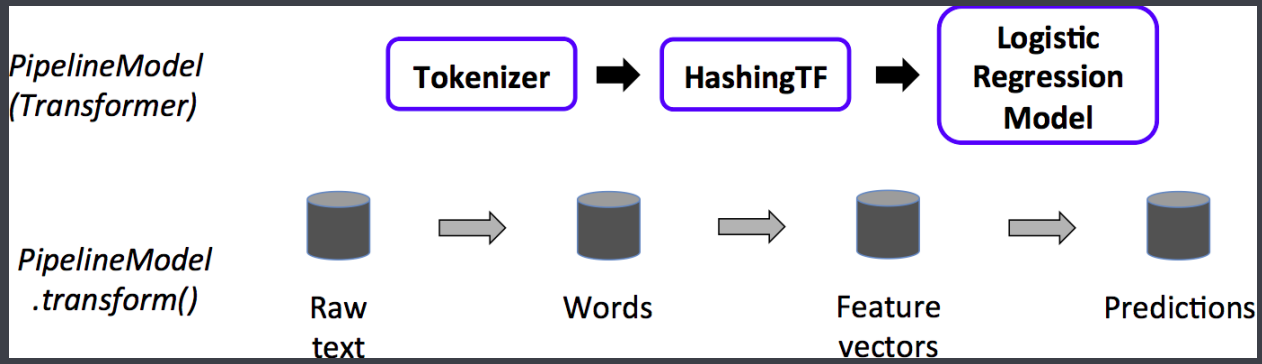
- Transformer: DF => another DF
- Estimator: DF => model
- Model: DF => DF with predictions
- Pipeline: DF => [transformer/estimator] => DF
- Evaluator: DF => double



Entrenamiento Multi-modelo:

- *Estimators:* independientes de los modelos que producen. Así se puede optimizar el entrenamiento con varios modelos
  - computar multiples gradientes en una sóla pasada
  - replicar el dataset y entrenar de manera separada.

def fit(df: DataFrame, paramMaps: Array[ParamMap]): Array[Model]



Configuración de Parámetros

- **Parámetros: pueden ser referenciados fuera del ámbito del algoritmo (type-safe).**

```
val paramGrid = new ParamGridBuilder()
  .addGrid(hashingTF.numFeatures, Array(1000, 10000))
  .addGrid(lr.regParam, Array(0.05, 0.1, 0.2))
  .build()
```

- **Validación-cruzada: independiente al estimador, evaluador y a los hiper-parámetros.**

```
val cv = new CrossValidator()
  .setEstimator(pipeline)
  .setEvaluator(evaluator)
  .setEstimatorParamMaps(paramGrid)
  .setNumFolds(3)
```

Novedades en MLLIB (1.3 – 1.6)

Atributos ML:

- Tipos diferentes: numérico, nominal y binario (DF).
- Estádisticas de resúmen.

Feature transformes:

- Label indexer, one-hot encoder, etc.

Más algoritmos en la pipeline API (spark.ml)

Persistencia del pipeline (New!)

Integración con SparkR.

Dataset API (New!) - A new Spark API, similar to RDDs, that allows users to work with **custom objects and lambda functions** while still gaining the benefits of the Spark SQL execution engine.

Recursos

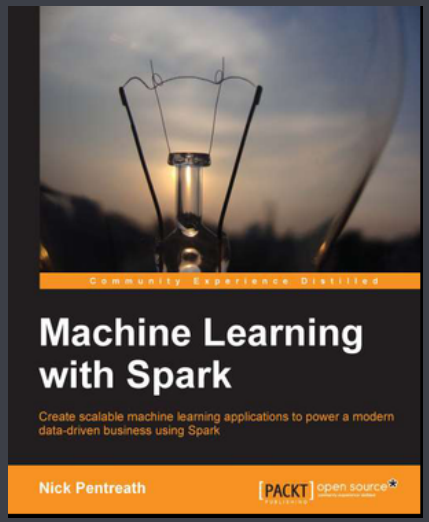
Documentación oficial MLLIB: http://spark.apache.org/docs/latest/mllib-guide.html

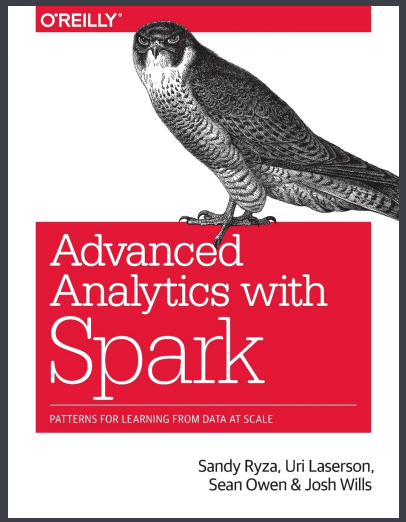
Spark packages: https://spark-packages.org/

Blog Databricks: https://databricks.com/blog

Paper MLLIB: http://jmlr.org/papers/v17/15-237.html

Algunos libros:





Thanks!

ANY QUESTIONS?

You can find me at sramirez@decsai.ugr.es

CREDITS

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash