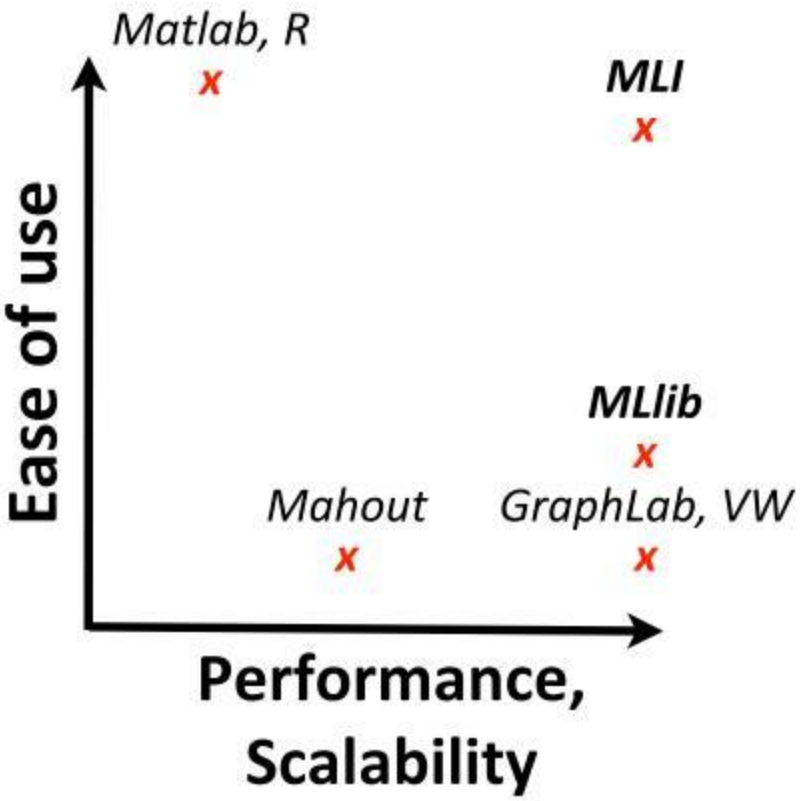
Scalable Machine Learning for Spark

Sam Bessalah - @samklr

Lay of the Land



MATLAB

+ Easy (Resembles math, limited set up)
+ Sufficient for prototyping / writing papers
— Ad-hoc, non-scalable scripts
— Loss of translation upon re-implementation

+ Scalable and (sometimes) fast
+ Existing open-source libraries
— Difficult to set up, extend

mahout

VOWPAL WABBIT

GraphLab

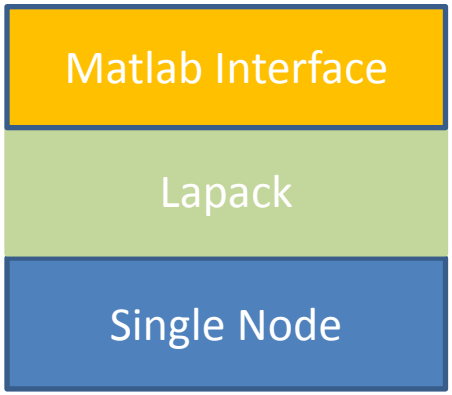
1. Easy scalable ML development (*ML Developers*)
2. User-friendly ML at scale (*End Users*)

Along the way, we gain insight into data intensive computing

MATLAB Example

- Matlab has an integrated environment for ML development.
- Data access and processing abstractions
- Leverages and extends the Lapack functionalities

But doesn't scale to distributed environments.



Spark : In memory, fast cluster computing system suited for iterative computations like Machine learning algortihms.

Spark : In memory, fast cluster computing system

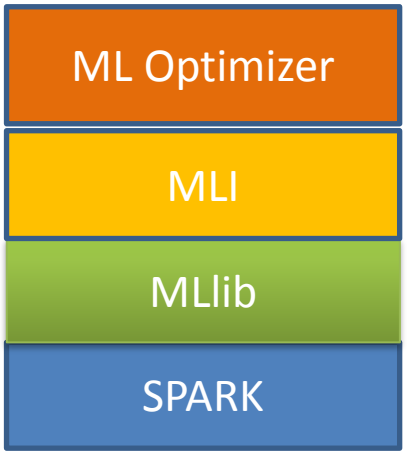
MLLib : Low level ML algorithms implemented as Spark's standard ML library

Spark : In memory, fast cluster computing system

MLLib : Low level ML algorithms in SPARK.

MLI : API / Platform for feature extraction, data pre processing and algorithm consumption. Aims to be platform independent.

MLBASE



Spark : In memory, fast cluster computing system

MLlib : Low level ML algorithms in Spark.

MLI : API / Platform for feature extraction, data pre processing and algorithm consumption. Aims to be platform independent.

ML Optimizer : automates model selection, by solving a search problem over features extractors

MLlib

- Contains core algorithms implemented using Spark programming model for cluster computing : RDD
- So far contains algorithms for :
  - Regression : Ridge, Lasso
  - Classification : Support Vector Machine, Logistic Regression,
  - RecSys : Matrix Factorisation with ALS
  - Clustering : K means
  - Optimisation : Stochastic Gradient Descent

More being contributed ….

MLI : ML developer API

- Aims to shield ML developers from runtimes implementation.

- Provides high level abstractions and operators to build models in a distributed environment, along with data processing tools.

- Linear Algebra : **MLMatrix, MLRow, MLVector …**
  - Linear algebra on local partitions
  - Sparse and Dense matrix support

- Table Computations : **MLTable**
  - Similar to R/Pandas(Python) DataFrame or NumPy Array
  - Flexibility when loading /processing data
  - Common interface for feature extraction

Number of lines of Code

Logistic Regression

System	Lines of Code
Matlab	11
Vowpal Wabbit	721
MLI	55

Alternating Least Squares

System	Lines of Code
Matlab	20
Mahout	865
GraphLab	383
MLI	32

MLI

- Functionalities :

  -Regression : Linear Regression (Ridge, Lasso)

  -Classification : Logistic Regression, Support Vector Machines, Decision Trees, Naive Bayes

  -Collaborative Filtering : Alternating Least Squares, DFC

  -Clustering : K-means, DP-Means

  -Feature Extraction : PCA, N-grams, feature normalizer, Vectorizers

  -Optimization : Parallel Gradient, Local SGD, L-BFGS

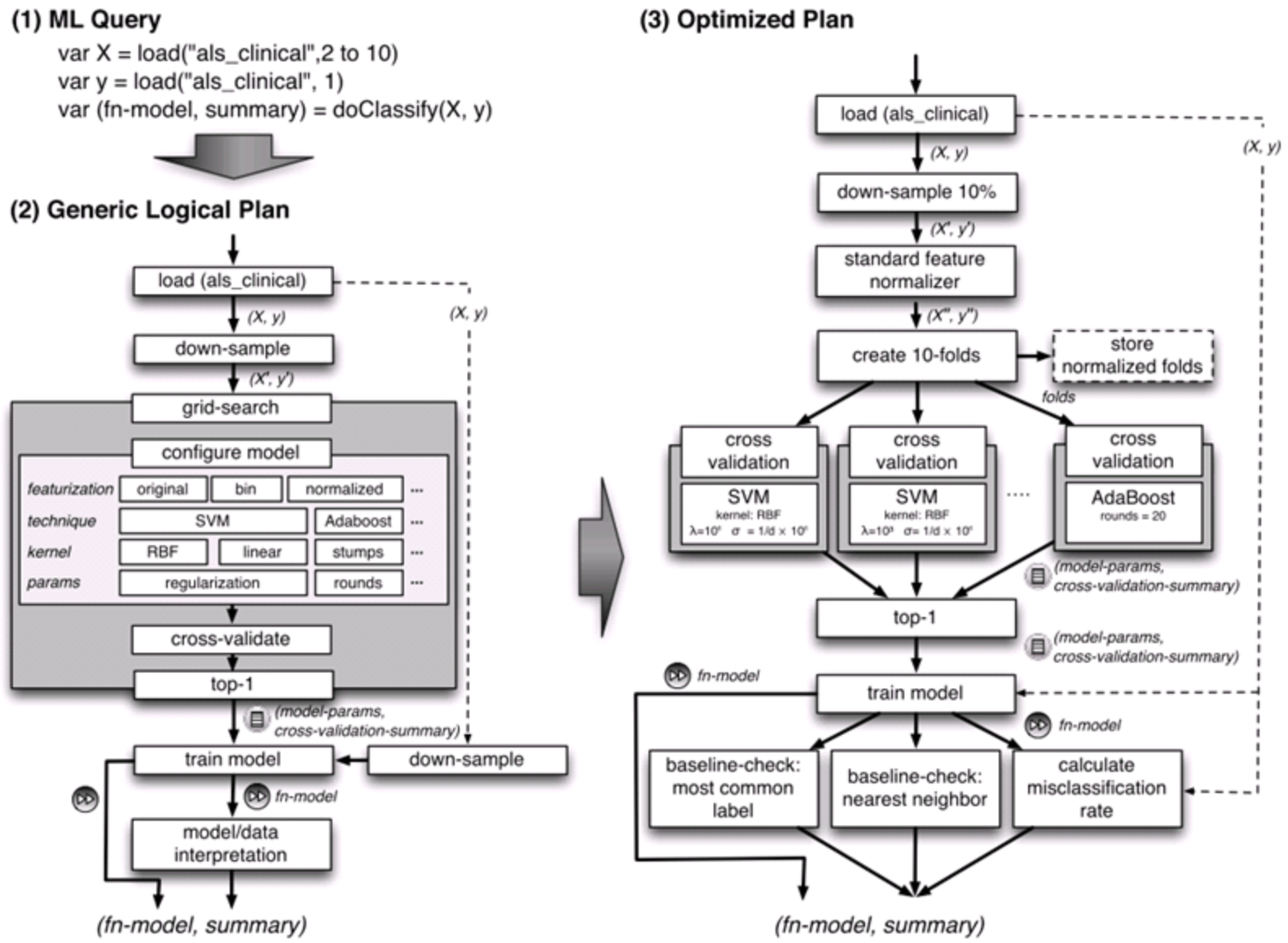
  -Model evaluation : Cross validation, Evaluation metrics

ML Optimizer

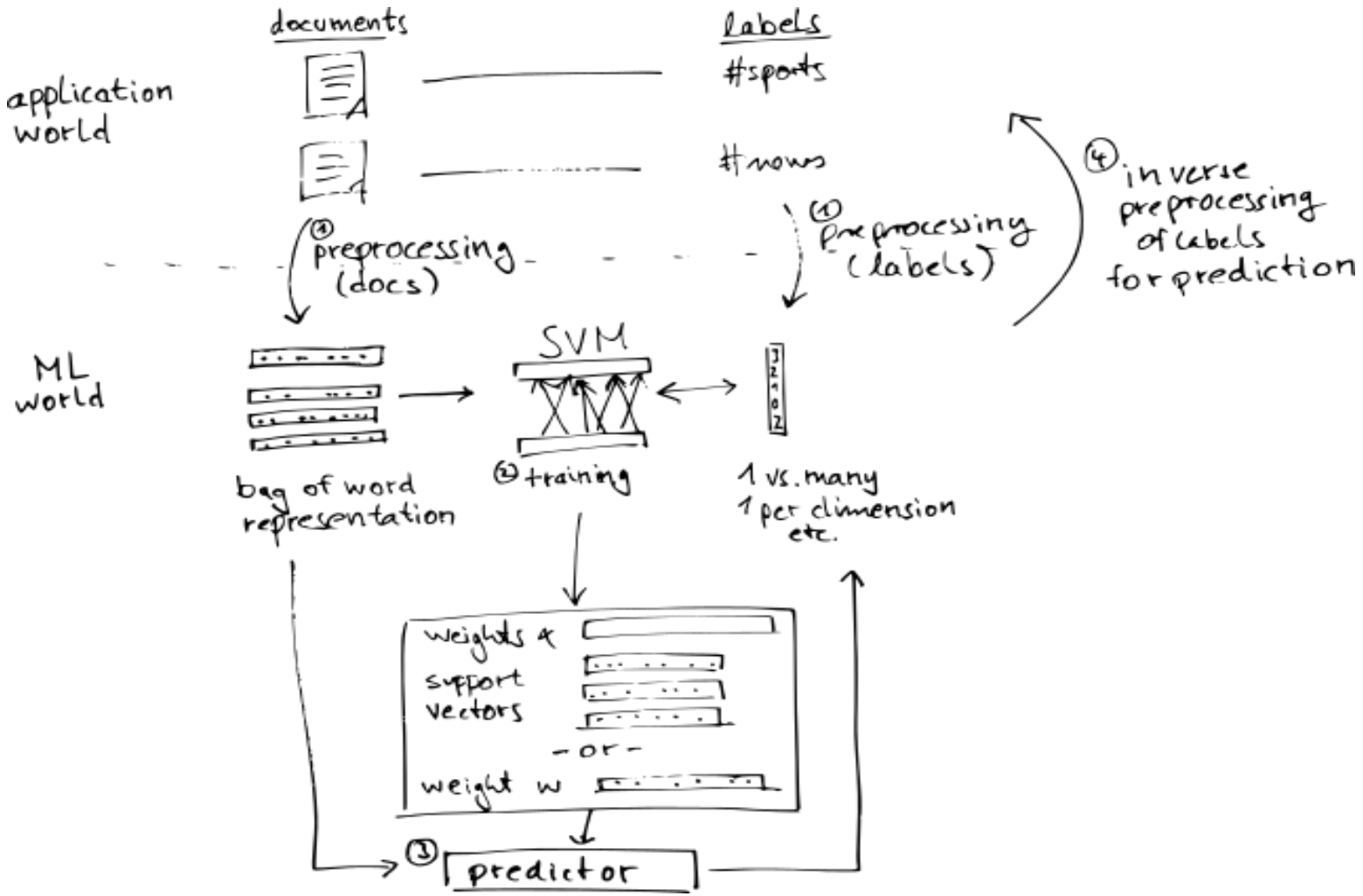
- Aimed at non ML developers
- Specify ML tasks declaratively
- Have the system do all the heavy lifting using MLI and MLLib.

```
var X = load (''local_file '', 2 to 10)
var Y = load (''text_file'', 1)
var  (fn-model, summary) = doClassify(X, y)
```

Not available yet, currently under development .


(1) ML Query
var X = load("als_clinical",2 to 10)
var y = load("als_clinical", 1)
var (fn-model, summary) = doClassify(X, y)
(2) Generic Logical Plan
load (als_clinical)
(X, y)
down-sample
(X', y')
grid-search
configure model
featurization
original
bin
normalized
technique
SVM
Adaboost
kernel
RBF
linear
stumps
params
regularization
rounds
cross-validate
top-1
(model-params, cross-validation-summary)
train model
down-sample
fn-model
model/data interpretation
(fn-model, summary)
(3) Optimized Plan
load (als_clinical)
down-sample 10%
standard feature normalizer
create 10-folds
store normalized folds
folds
cross validation
SVM
kernel: RBF
AdaBoost
rounds = 20
top-1
train model
baseline-check: most common label
baseline-check: nearest neighbor
calculate misclassification rate
(fn-model, summary)

Example Worklow : Document Classification



Typical data analysis workflow

Spark /MLI

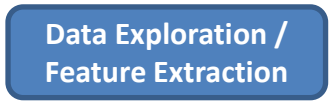
load.scala Scala

```scala

val inputTable = mc.loadFile("/enwiki_txt")
                   .filter(r => List("ARTS","LIFE")
                   .contains(r(0).toString)
                   .cache()

val firstFive = inputTable.take(5)
val taggedInputTable = inputTable.project(Seq(0,2))
                                 .map(r => {
                                       val label = if(r(0).toString == "ARTS") 1.0 else -1.0
                                       MLRow(label, r(1))
                                 }).cache()
```

Typical data analysis workflow

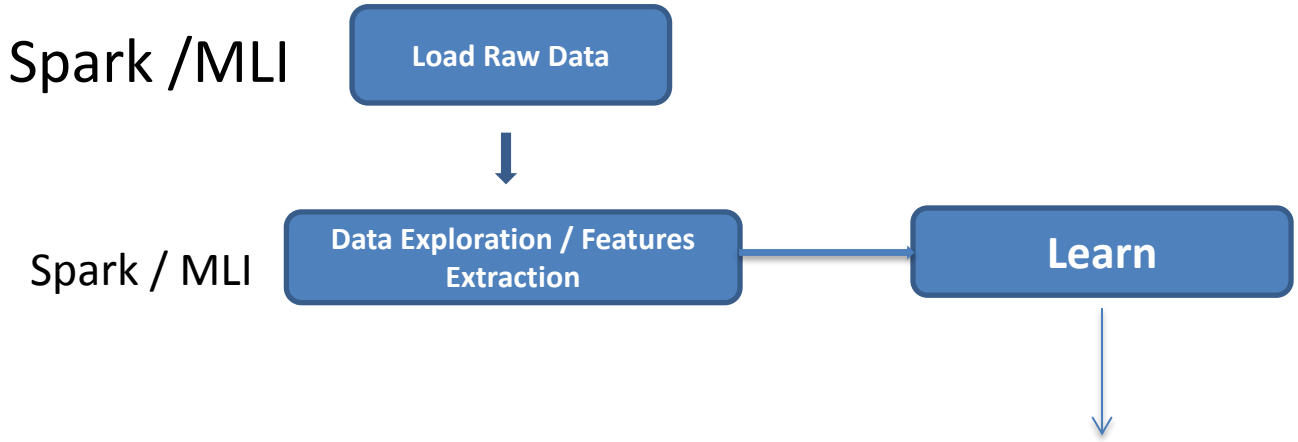
Spark /MLI

Spark / MLI



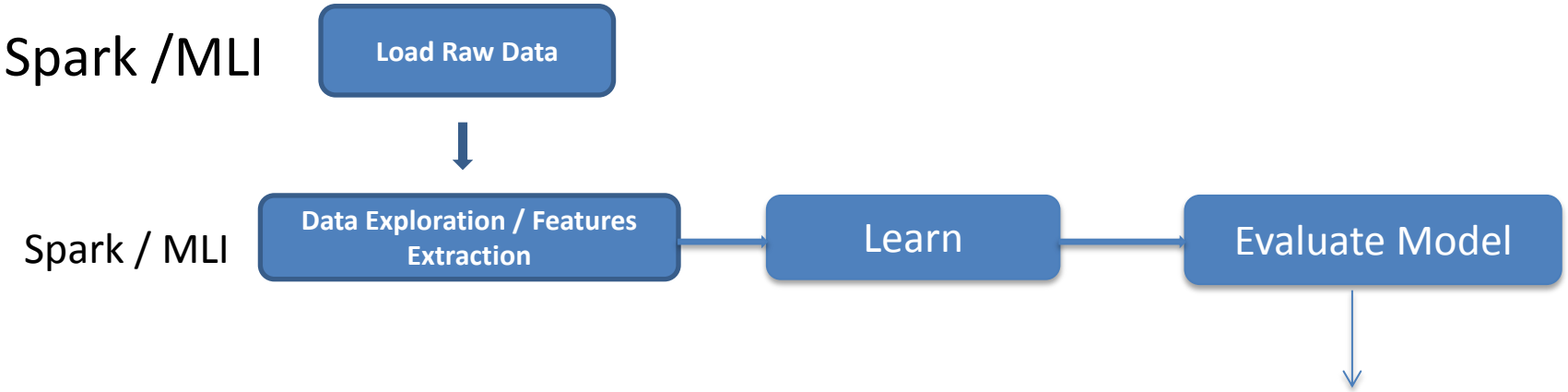

```
import mli.feat._
// c is the column on which we want to perform N-gram extraction
// n is the N-gram length, e.g., n=2 corresponds to bigrams
// k is the number of top N-grams we want to use (sorted by N-gram frequency)
val (featurizedData, ngfeaturizer) = NGrams.extractNGrams(taggedInputTable, c=1, n=2, k=1000,
                                                          stopWords = NGrams.stopWords)

val (scaledData, featurizer) = Scale.scale(
                                           featurizedData.filter(_.nonZeros.length > 5).cache(),
                                           0,
                                           ngfeaturizer
                                          )
```

Typical data analysis workflow



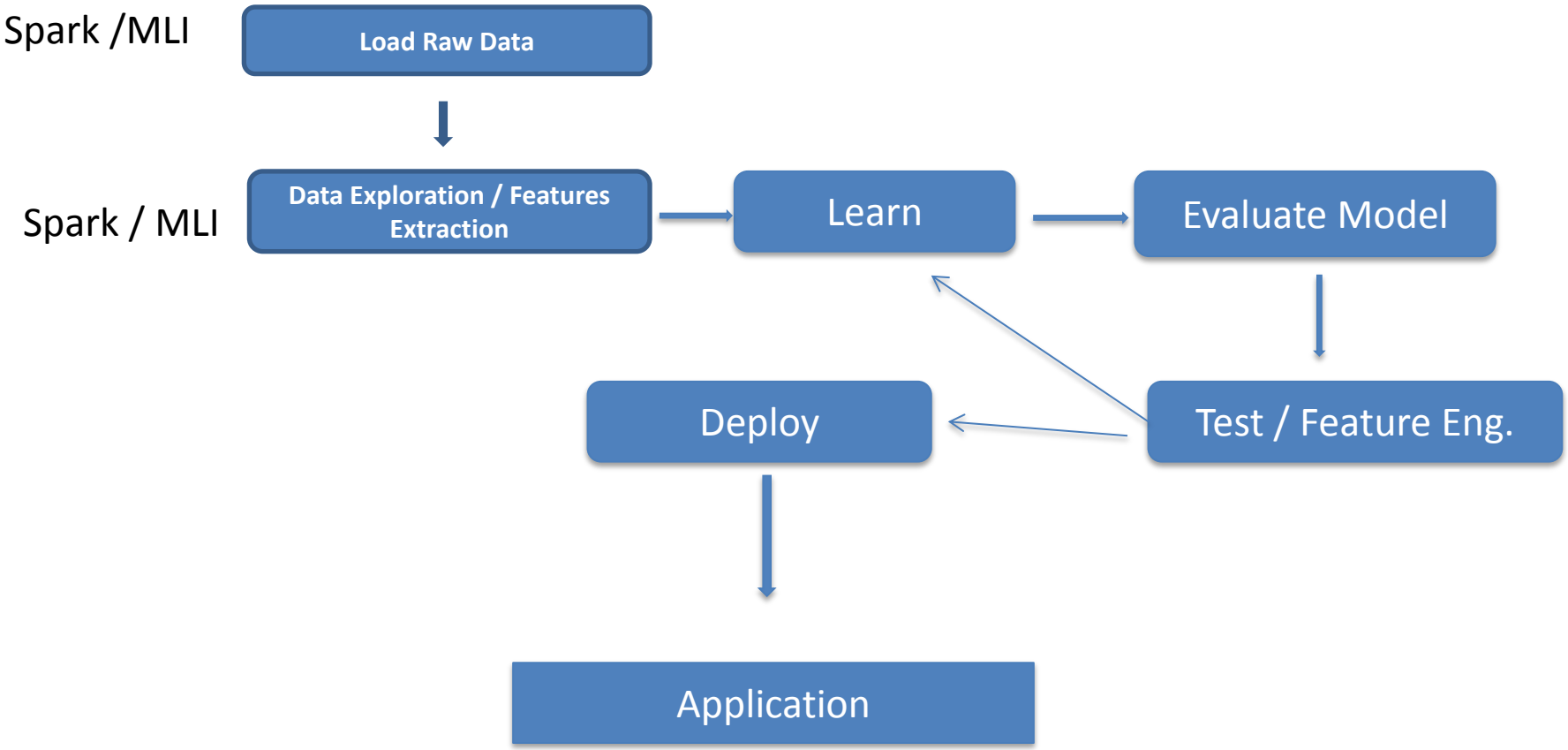

```
import mli.ml.classification._
val model = SVMAlgorithm.train(trainData, SVMParameters(learningRate=10.0,regParam=1.0, maxIterations=50))

 //Prediction + Model assessment
val firstDataPoint = trainData.take(1)(0)
model.predict(firstDataPoint.tail)
val trainVsPred = trainData.map(r => MLRow(r(0), model.predict(r.tail)))
val trainError = trainVsPred.filter(r => r(0) != r(1)).numRows.toDouble/trainData.numRows
```

Typical data analysis workflow

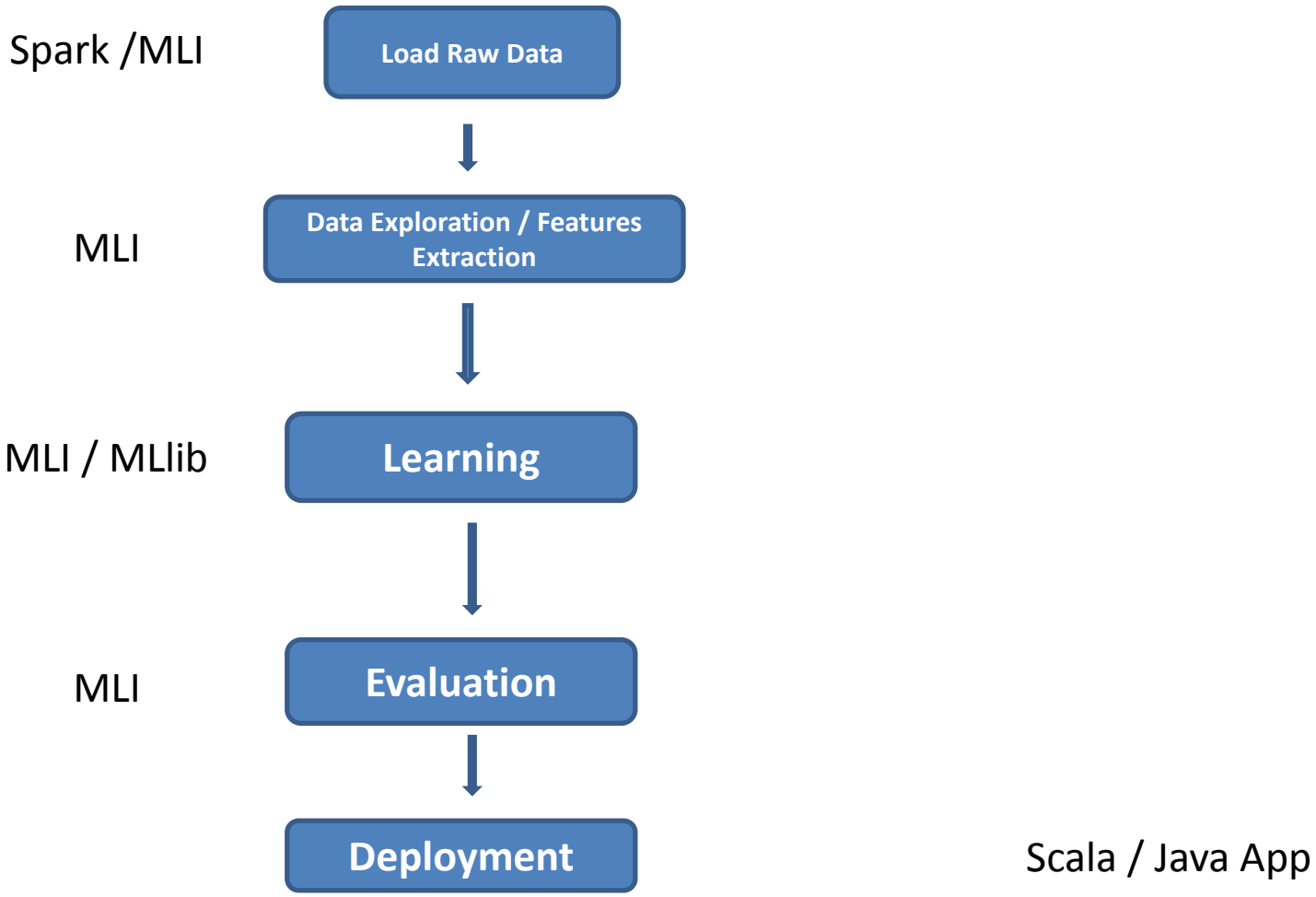



```
def evalModel(model: SVMModel, testData: MLTable) = {
val trainData = model.trainingData
val trainVsPred = trainData.map(r => MLRow(r(0), model.predict(r.tail)))
val trainErr = trainVsPred.filter(r => r(0).toNumber != r(1).toNumber).numRows.toDouble / trainData.numRows
val testVsPred = testData.map(r => MLRow(r(0), model.predict(r.tail)))
val testErr = testVsPred.filter(r => r(0).toNumber != r(1).toNumber).numRows.toDouble / testData.numRows
(trainErr, testErr)
}

// Features engineering.
val topFeatures = model.features.sortWith(_._2 < _._2).take(10)
val bottomFeatures = model.features.sortWith(_._2 > _._2).take(10)
```

Typical data analysis workflow



Typical data analysis workflow



Current status

- MLlib is avilable in Spark from the release 0.8

  https://github.com/apache/incubator-spark/tree/master/mllib
- MLI is developed separately

  http://github.com/amplab/MLI
- ML Optimizer, is an active research project in common with the AmpLab Berkeley, VmWAre, and Brown University, first release planned for winter.
- Looking for contributors , very welcoming community ☺

http://spark.incubator.apache.org