Machine learning with Apache Spark

Petar Zečević

petar.zecevic@svgroup.hr

@p_zecevic

Apache Spark Zagreb Meetup group

http://www.meetup.com/Apache-Spark-Zagreb-Meetup

39% off with the code "zecevic39"

May 20, half off with "dotd052016au"

http://bit.ly/sparkinaction

Agenda for today

What is Spark?

How does Spark work?

About machine learning

Machine learning with Spark

Spark and other ML frameworks

Q & A

What is Apache Spark?

It is a ...

Show of hands

- I've never used Apache Spark
- I've played around with it
- I'm planning to or I'm already using Spark in production

What is Apache Spark?

It is a ...

distributed

general-purpose

large-scale

data processing engine

Distributed

Runs on many machines

Runs in a cluster

Parallelizes computations

Sends program to data

General-purpose

Used across industries

Reads any kind of data

Writes any kind of data

Does whatever Java/Python/R can do

Large-scale

Big Data: 

But also: Banks Healthcare

Biology Physics

Weather Astrophysics

But you should have enough data

large-scale

data processing engine

Data processing engine

It's about processing data

It's not your ERP system

It's not your web framework

It's not for rendering videos

Why Spark?

(Why not MapReduce)

Spark is fast

- Works mostly in memory
- Especially fast for iterative tasks
- 100 times faster than MapReduce
- Broke sorting world record in 2014
- Project Tungsten brought additional optimizations in v1.5

Simple and concise API

- Functional programming
- Distributed collections feel like local ones
- APIs in Scala, Java, Python and R

Simple and concise API

MAPREDUCE WORD COUNT

Main class

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;

public class WordCount
{
      public static void main(String[] args) throws Exception
      {
            JobConf conf = new JobConf(WordCount.class);
            conf.setJobName("Hadoop wordcount");
            conf.setOutputKeyClass(Text.class);
            conf.setOutputValueClass(IntWritable.class);
            conf.setMapperClass(WordCountMap.class);
            conf.setCombinerClass(WordCountReduce.class);
            conf.setReducerClass(WordCountReduce.class);
            conf.setInputFormat(TextInputFormat.class);
            conf.setOutputFormat(TextOutputFormat.class);
            FileInputFormat.setInputPaths(conf, new Path(args[0]));
            FileOutputFormat.setOutputPath(conf, new Path(args[1]));

            JobClient.runJob(conf);
      }
}
```

Mapper

```
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;

public class WordCountMap extends MapReduceBase
      implements Mapper<LongWritable, Text, Text, IntWritable>
{
      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();

      public void map(LongWritable key, Text value,
                  OutputCollector<Text, IntWritable> output, Reporter reporter)
            throws IOException
      {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
              word.set(tokenizer.nextToken());
              output.collect(word, one);
            }
      }
}
```

Reducer

```
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;

public class WordCountReduce extends MapReduceBase
      implements Reducer<Text, IntWritable, Text, IntWritable>
{
      public void reduce(Text key, Iterator<IntWritable> values,
                  OutputCollector<Text, IntWritable> output, Reporter reporter)
            throws IOException
      {
            int sum = 0;
            while (values.hasNext())
            {
                  sum += values.next().get();
            }
            output.collect(key, new IntWritable(sum));
      }
}
```

SPARK WORD COUNT

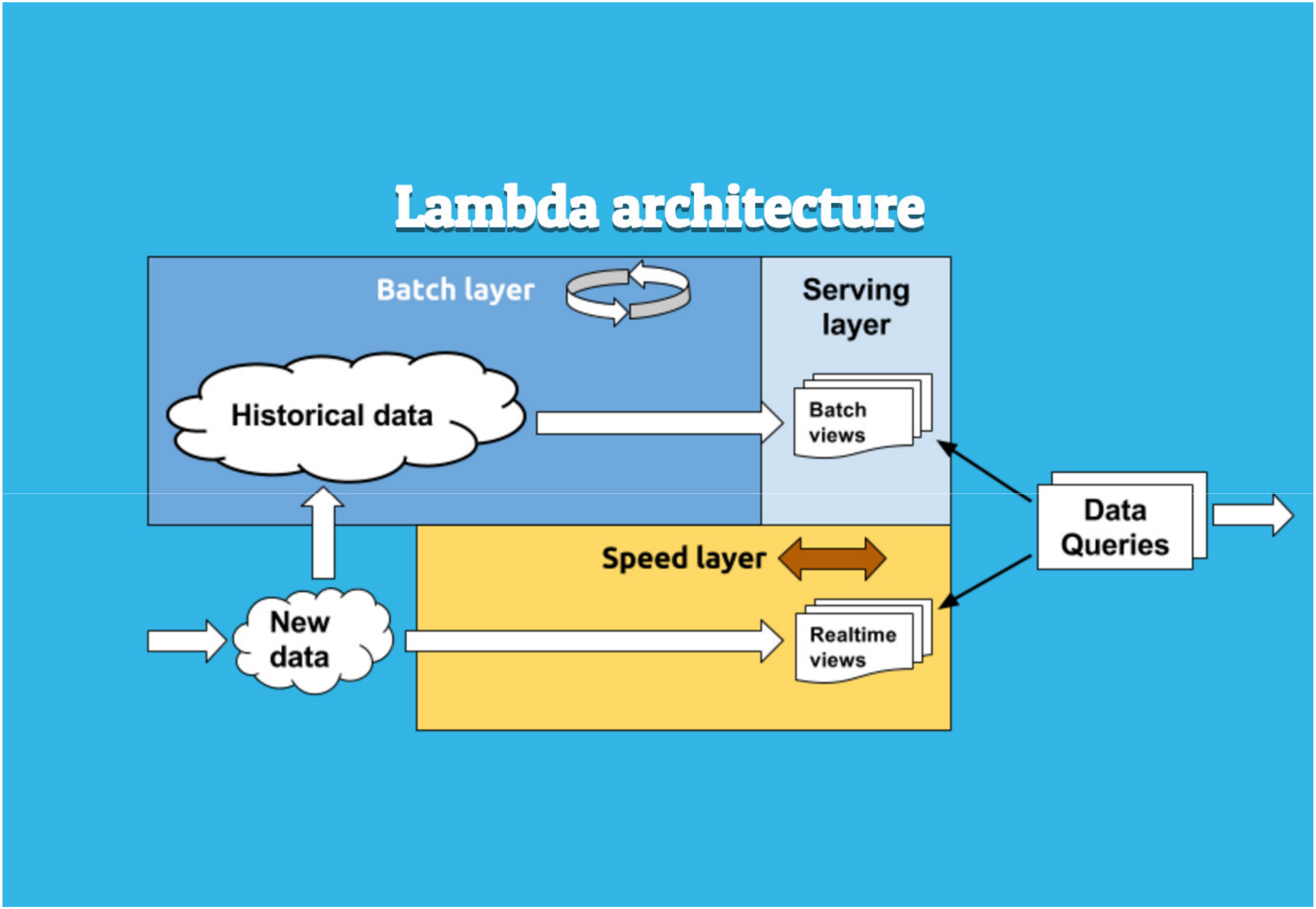
```
import org.apache.spark.{SparkConf, SparkContext}

object SparkWordCount
{
      def main(args : Array[String])
      {
            val conf = new SparkConf().setAppName("Spark wordcount")
            val sc = new SparkContext(conf)
            val file = sc.textFile("hdfs://...")
            val counts = file.flatMap(line => line.split(" "))
                .map(word => (word, 1))
                .reduceByKey(_ + _)
            counts.saveAsTextFile("hdfs://...")
      }
}
```

Spark is a unifying platform

In the same framework:

- Batch processing
- Real-time processing
- Analytics using SQL
- Machine learning
- Graph algorithms


Lambda architecture
Batch layer
Serving layer
Historical data
Batch views
Data Queries
Speed layer
New data
Realtime views

Spark has gone mainstream

- Large community
- Used in many companies
- In all major Hadoop distributions
- Many packages and connectors
- Conferences and certifications
- IBM recently pledged to commit 3000 developers to Spark

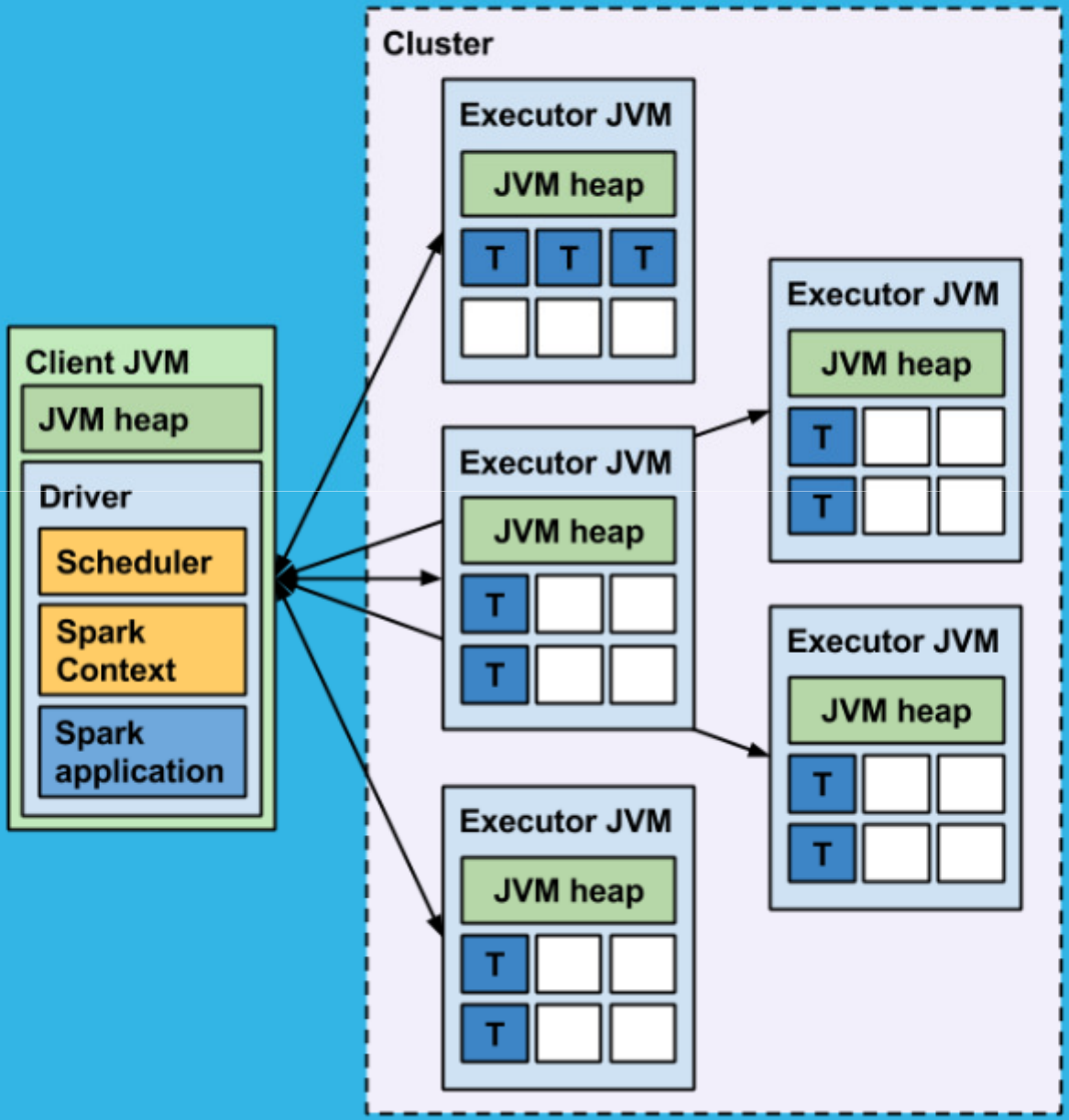
How do you use it?

- Install on commodity hardware
- Configure and start the cluster
- Write an application
- Execute application on the cluster
  (or use Spark shell)
- But not just Spark cluster:

  also runs on YARN

  also runs on Mesos

Basic architecture



RDD

Resilient Distributed Dataset

Represents distributed collections

Transformations: create new RDDs

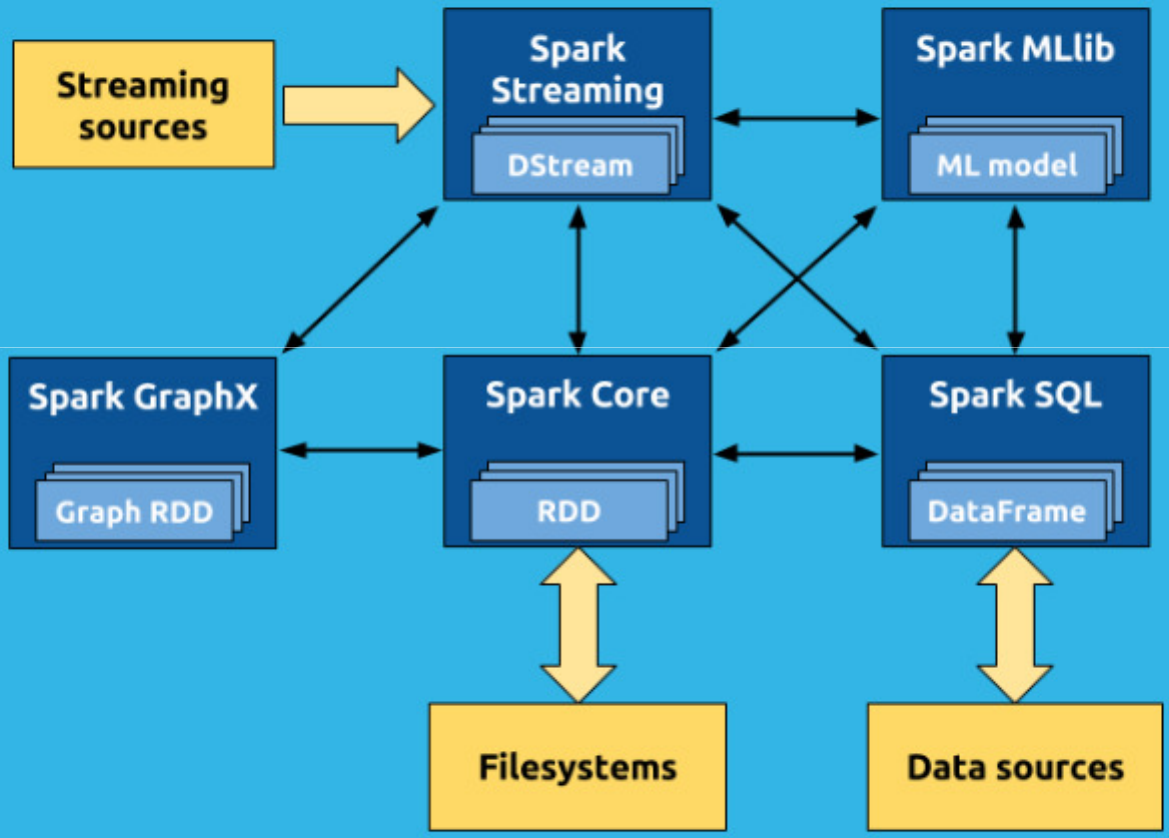
Actions: execute RDDs and get results

Example:

```
val rdd = sc.textFile("/some/file.txt")
val cnt = rdd.flatMap(line => line.split(" ")).
    map(word => (word, 1)).reduceByKey(_ + _)
cnt.collect()
```

What can you do with Spark?

Spark components



Spark Core

- Work with RDDs
- Load, save, parse raw data
- Transform, aggregate, join data sets
- Communicate with executors

Spark SQL

- Work with **DataFrames**
- Handle structured data, organized in columns
- Load/save structured data from/to external data sources
- Transform, aggregate, join data using SQL
- Query data through a JDBC server

Spark Streaming

- Handle real-time data
- Use the same API as batch jobs
- Latency from half a second and up
- Connectors to Kafka, Flume, etc.

Spark GraphX

- Vertices and edges
- Graph algorithms
- Page rank, connected components
- Triangle count, shortest paths

Spark ML

- Machine learning algorithms
- Classification, regression, clustering, PCA, TF-IDF, ...
- Neural networks under development

Machine learning with Spark

What is machine learning?

A scientific discipline which
explores the construction and study of
algorithms
that can learn from and make
predictions on data
(Ron Kohavi, Foster Provost, 1998)

What is machine learning?

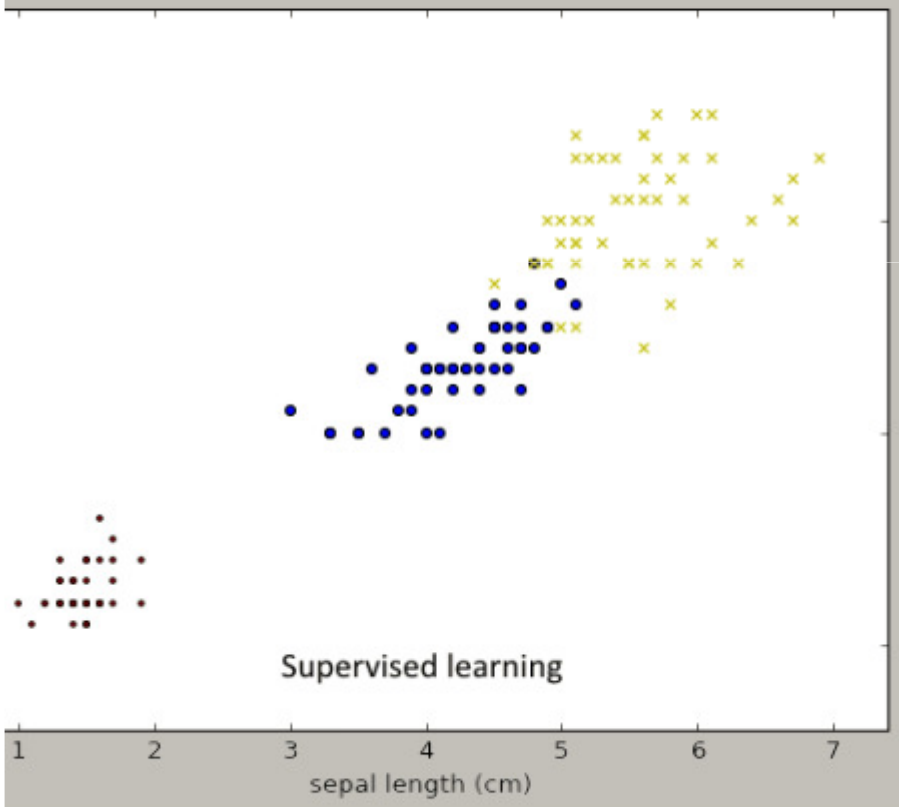
Relies on methods from the fields of

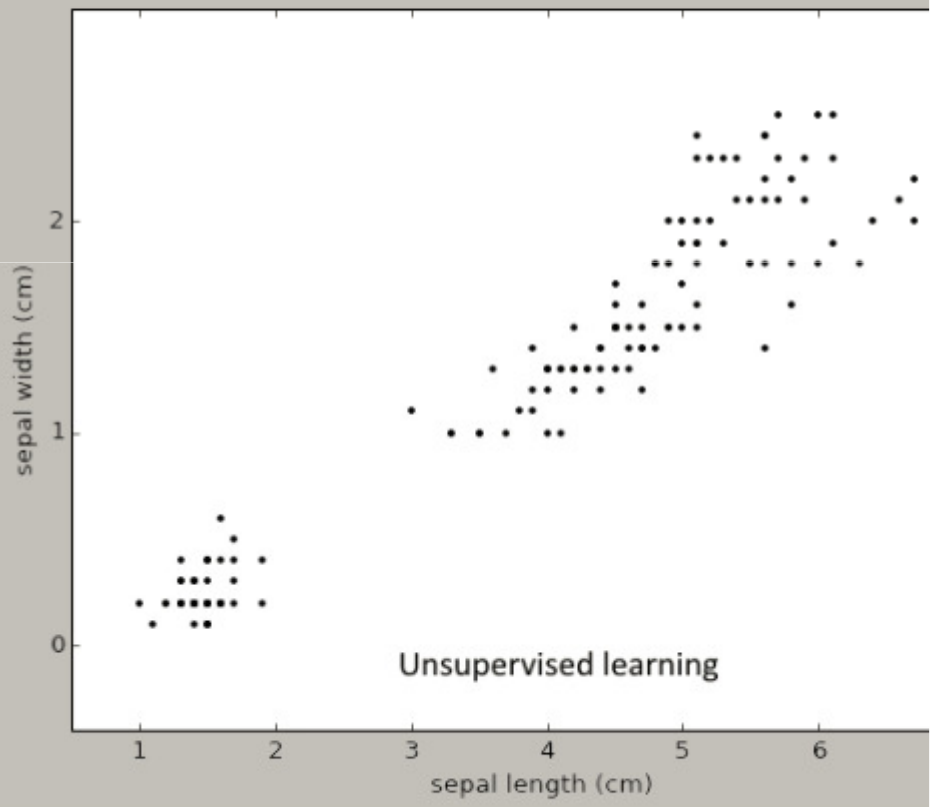
- statistics
- probability theory
- information theory

to discover and use the knowledge inherent in the data.

Machine learning algorithms

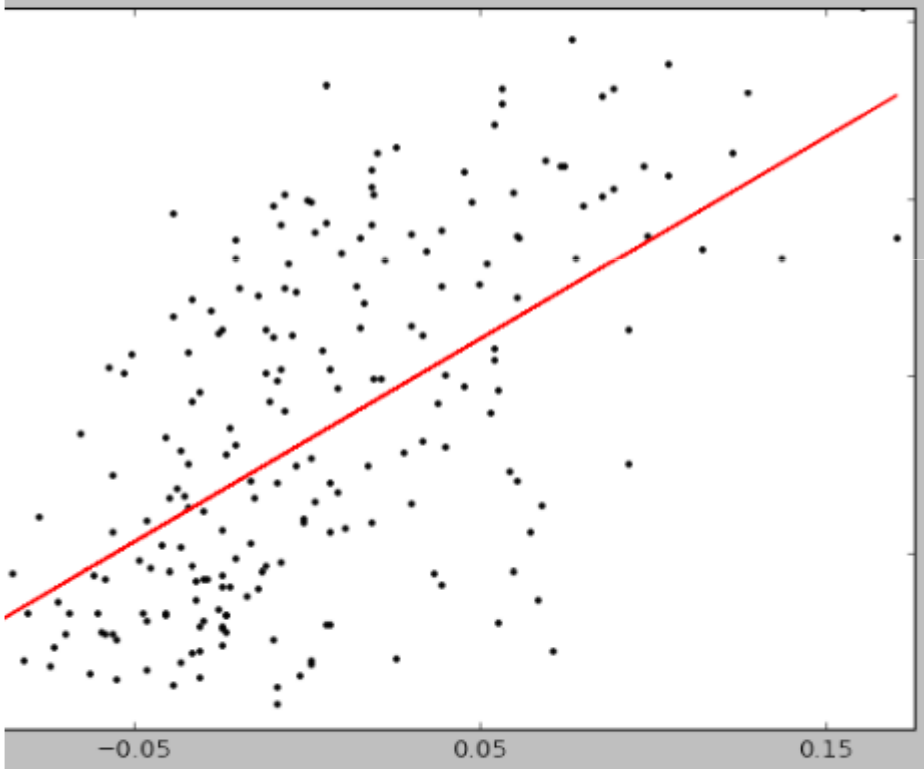
Supervised and unsupervised

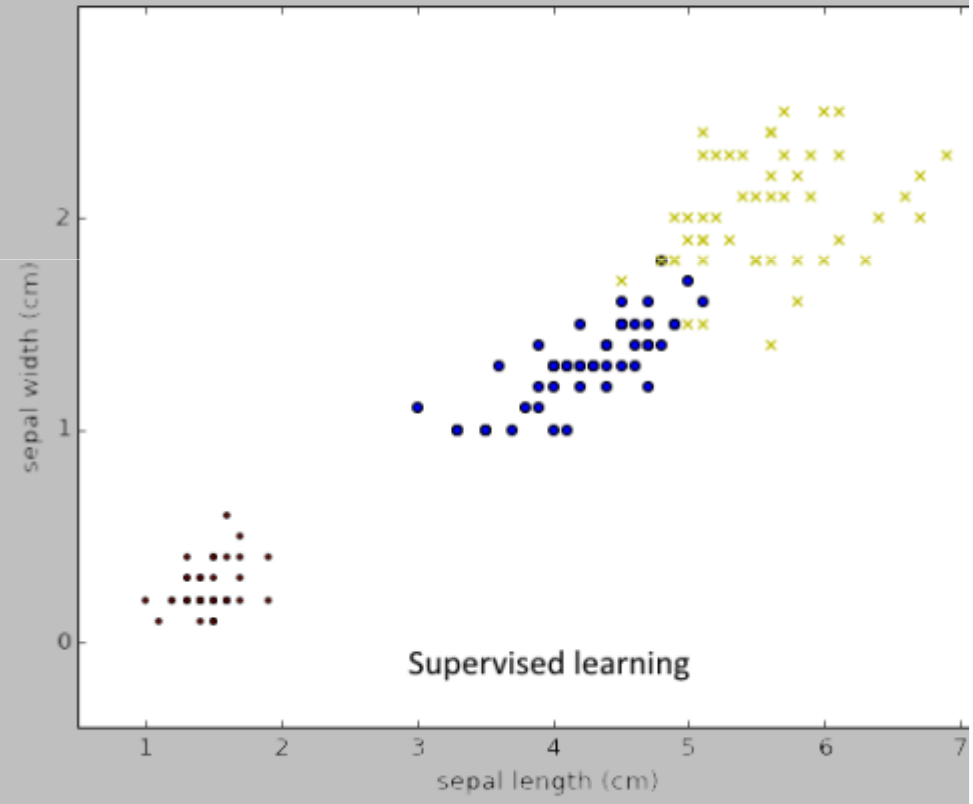




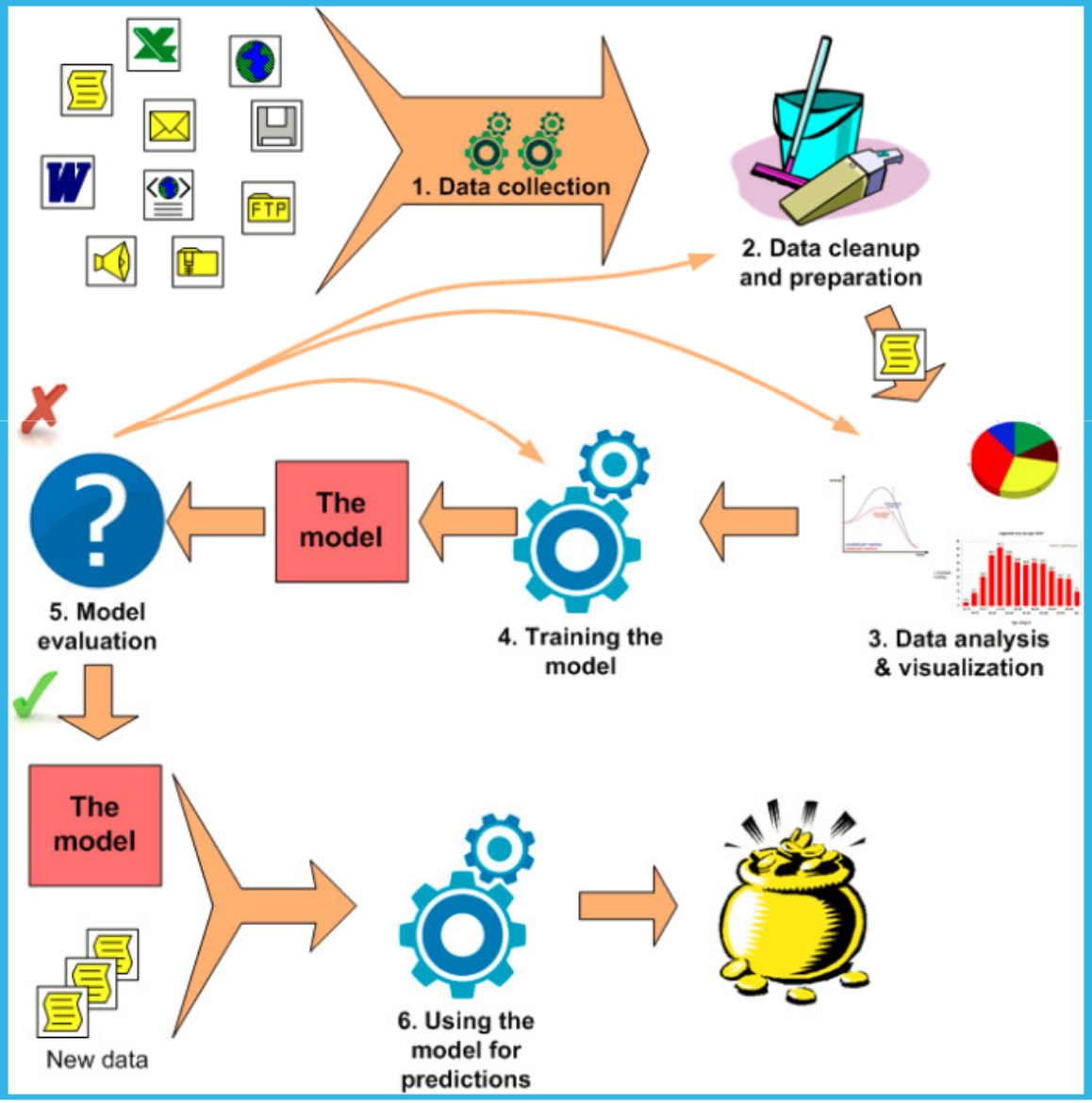
Machine learning algorithms

Regression and classification





ML is not just about training and using ML models



Two main sections of the Spark API

- Spark MLlib - the original Spark ML library
- Spark ML - new algorithm implementations with standardized ML pipelines

Spark MLlib

- RDD-based
- Distributed matrices and vectors
- 100 times faster than Map-reduce (iterations!)

Spark MLlib algorithms

- Feature transf., summary stats
- Principal component analysis
- Linear and logistic regression
- Support vector machines
- Naive Bayes
- Random forest
- Gradient-boosted trees
- K-means clustering
- …

MLlib API - splitting the data set

```
import org.apache.spark.mllib.regression.LabeledPoint
val data = rdd.map(x => {
  val a = x.toArray;
  LabeledPoint(a(a.length-1),
    Vectors.dense(a.slice(0, a.length-1)))
})
val sets = data.randomSplit(Array(0.8, 0.2))
val dataTrain = sets(0)
val dataValid = sets(1)
```

MLLib API - scaling data

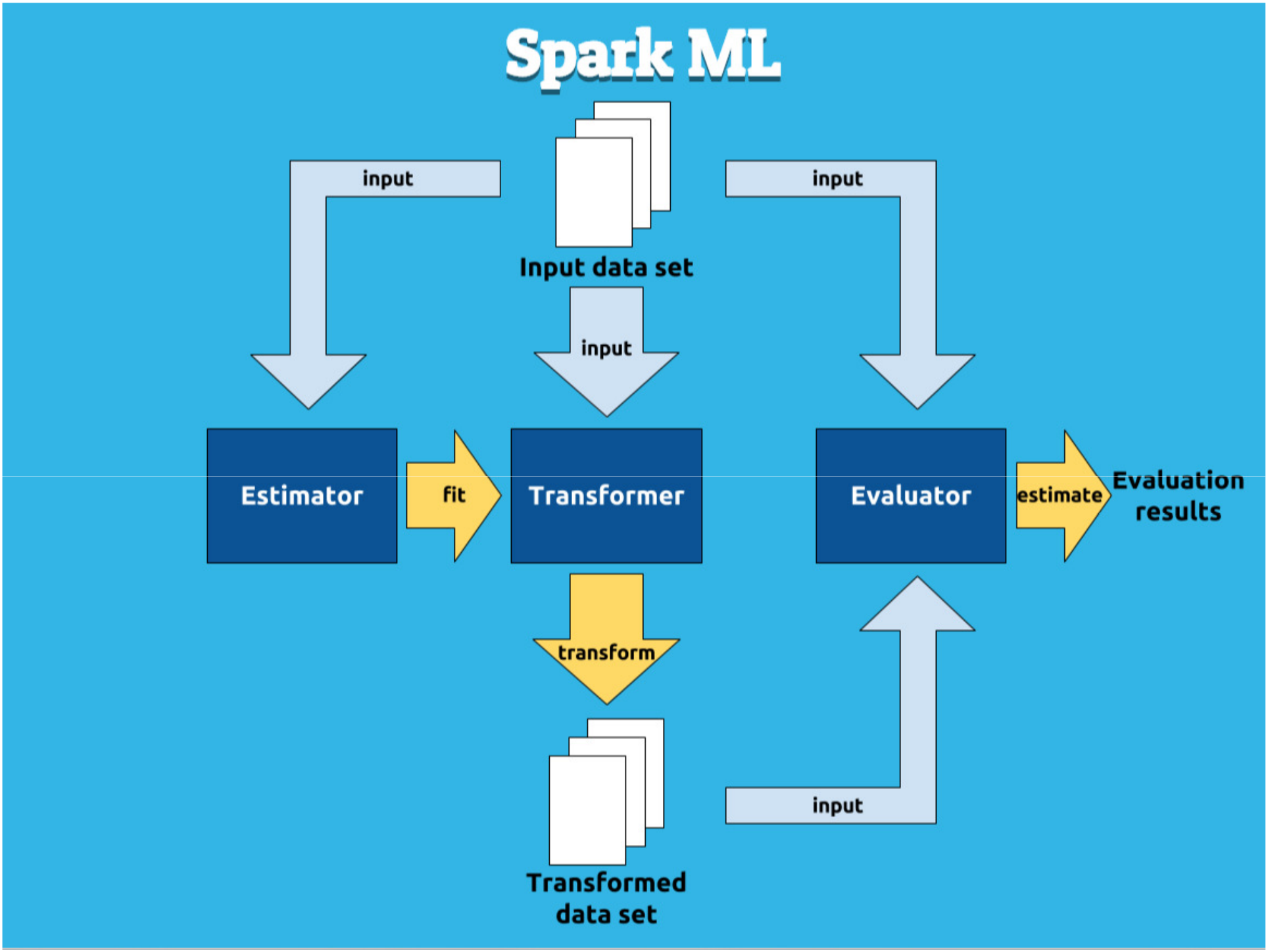
```
import org.apache.spark.mllib.feature.StandardScaler
val scaler = new StandardScaler(true, true).
  fit(dataTrain.map(x => x.features))
val trainScaled = dataTrain.map(x =>
  LabeledPoint(x.label, scaler.transform(x.features)))
val validScaled = dataValid.map(x =>
  LabeledPoint(x.label, scaler.transform(x.features)))
```

MLlib API - training a linear regression model

```
import org.apache.spark.mllib.regression.
  LinearRegressionWithSGD
val alg = new LinearRegressionWithSGD()
alg.setIntercept(true)
alg.optimizer.setNumIterations(200)
trainScaled.cache()
validScaled.cache()
val model = alg.run(trainScaled)
val validPredicts = validScaled.map(x =>
  (model.predict(x.features), x.label))
validPredicts.collect()
import org.apache.spark.mllib.evaluation.RegressionMetrics
val validMetrics = new RegressionMetrics(validPredicts)
validMetrics.rootMeanSquaredError
validMetrics.meanSquaredError
```

Spark ML

- DataFrame-based
- Tungsten performance improvements
- Machine learning pipelines standardized
- PipelineModel is just another Transformer


Spark ML
input
Input data set
input
input
Estimator
fit
Transformer
Evaluator
estimate
Evaluation results
transform
Transformed data set
input

Spark ML algorithms

- Feature transf., summary stats, one-hot encoding, string indexing
- Linear and logistic regression
- Random forest
- Naive Bayes
- Gradient-boosted trees
- K-means, Gaussian mixtures
- Alternating least squares
- …

ML API - training a linear regression model

```
val lr = new LogisticRegression
lr.setRegParam(0.01).setMaxIter(500).setFitIntercept(true)
val lrmodel = lr.fit(datatrain,
  ParamMap(lr.regParam -> 0.01,
  lr.maxIter -> 500, lr.fitIntercept -> true))
val validpredicts = lrmodel.transform(datavalid)
val bceval = new BinaryClassificationEvaluator()
bceval.setMetricName("areadUnderROC")
bceval.evaluate(validpredicts)
```

Other "compatible" frameworks

- Mahout
- H2O with Sparkling Water
- SparkNet (Caffe + Spark)
- DL4J
- XGboost4J

Want to know more?

http://www.meetup.com/Apache-Spark-Zagreb-Meetup

39% off !

Discount code: zecevic39

http://bit.ly/sparkinaction

Questions ?


THANK YOU
COME AGAIN
memegenerator.net