Face Recognition Model Using R in Spark Cluster

Zakaria Dahani, Yves Zango, Naima Touloum

MSc Big Data

Programming with Big Data in R using Distributed or Shared Memory

Outline

- Introduction
- Infrastructure - Cluster Deploiement
- Face recognition Model
- Conclusion
- Q&A

Introduction

A brief introduction 1/2

Divided in several steps:

- Find a topic theme and delimited it
- Realise the project

Objective

- Build a machine learning system on a Hadoop/Spark cluster
- Type of machine learning : face recognition

A brief introduction 2/2

Dataset :

- Training dataset: PubFig + LFW (fusion of 2 data sets)
- 35,469 training images and 11,720 test images
- 200 persons to identify

Infrastructure - Cluster Deploiement

Appache Spark

Apache Spark™ is a fast and general engine for large-scale data processing.

http://spark.apache.org/

What is

Speed

- Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Ease of Use

- Write applications quickly in Java, Scala, Python, R.

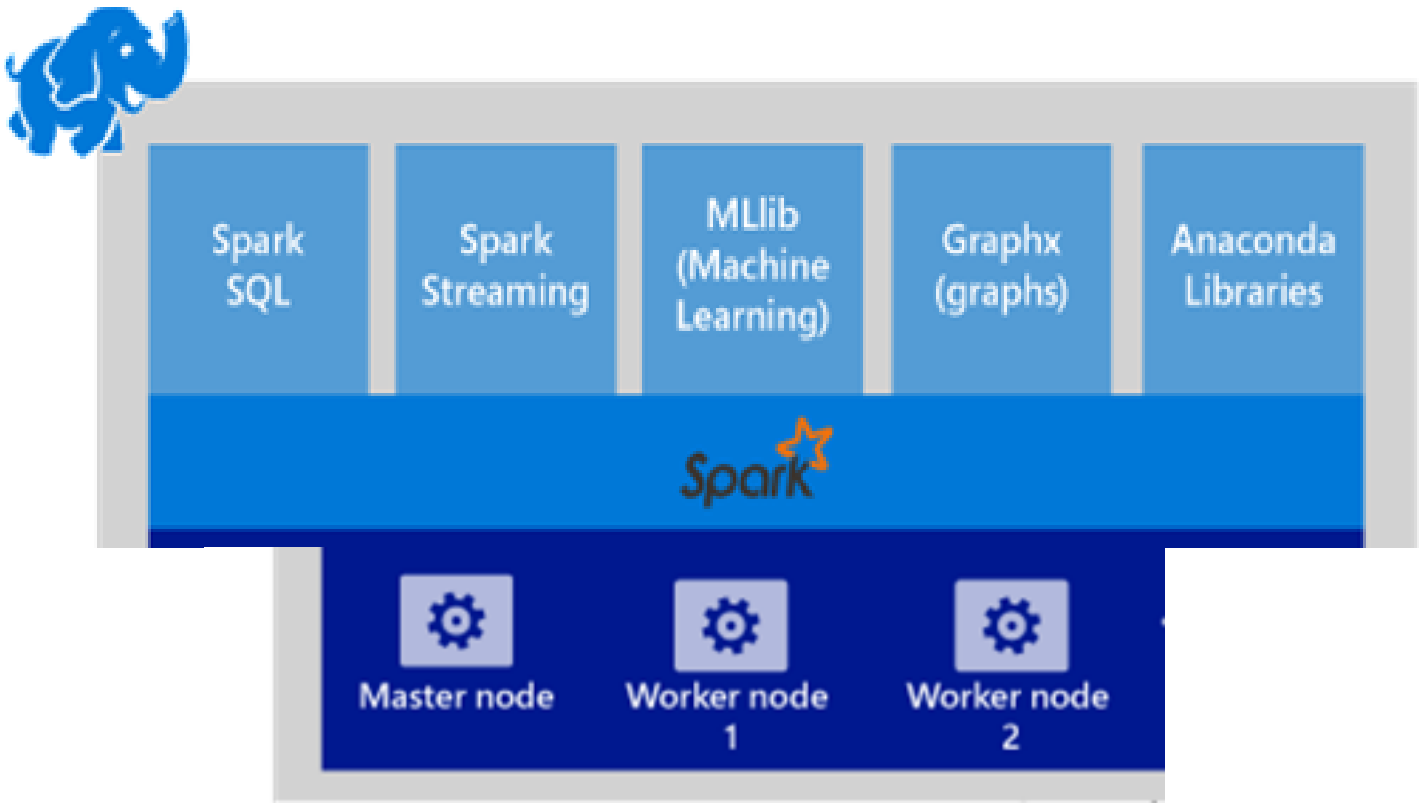
Generality

- Combine SQL, streaming, and complex analytics.

Runs Everywhere

- Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3.

Spark cluster in Hadoop



- 3 nodes with Hadoop 2.7 installed and configured
- 3 nodes with Spark 1.6 installed and configured
- SSH for passwordless communication between the master and the slaves

Sparklyr

R interface to Apache Spark, This package supports connecting to local and remote Apache Spark clusters, provides a 'dplyr' compatible back-end, and provides an interface to Spark's built-in machine learning algorithms.

https://cran.r-project.org/web/packages/sparklyr/index.html

What is

1

- New package that provides an interface between R and Apache Spark.
- Is NOT a replacement to the original apache spark but an extension to it.

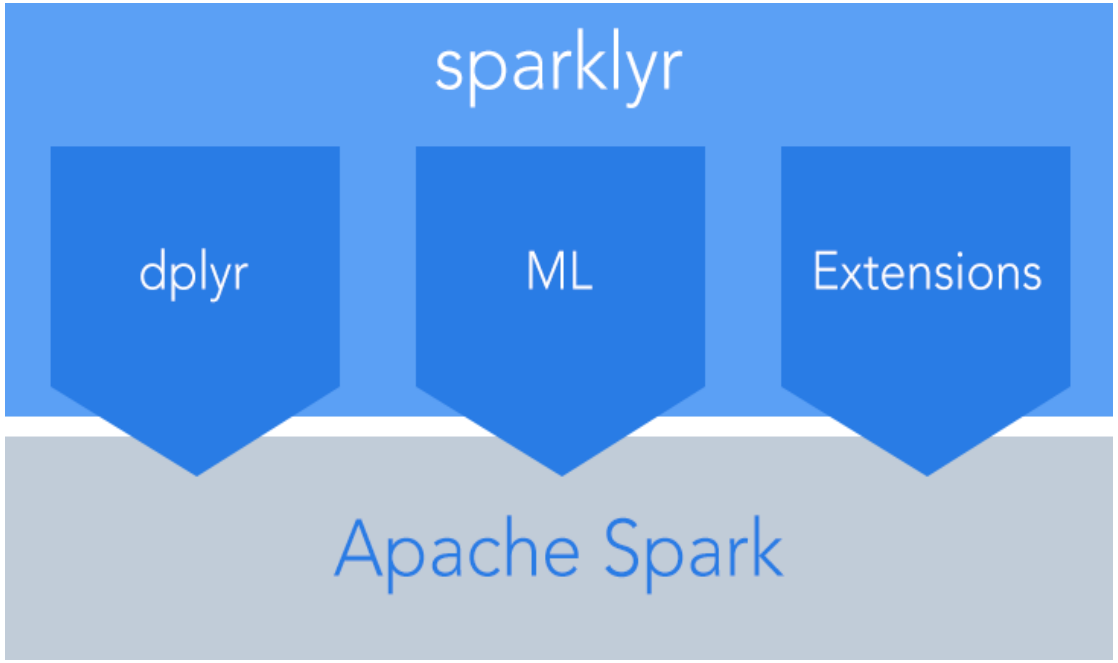
2

- Interactively manipulate Spark data using both dplyr and SQL
- Filter and aggregate Spark datasets then bring them into R for analysis and visualization.

3

- Create extensions that call the full Spark API and provide interfaces to Spark packages.

sparklyr: R interface for Apache Spark



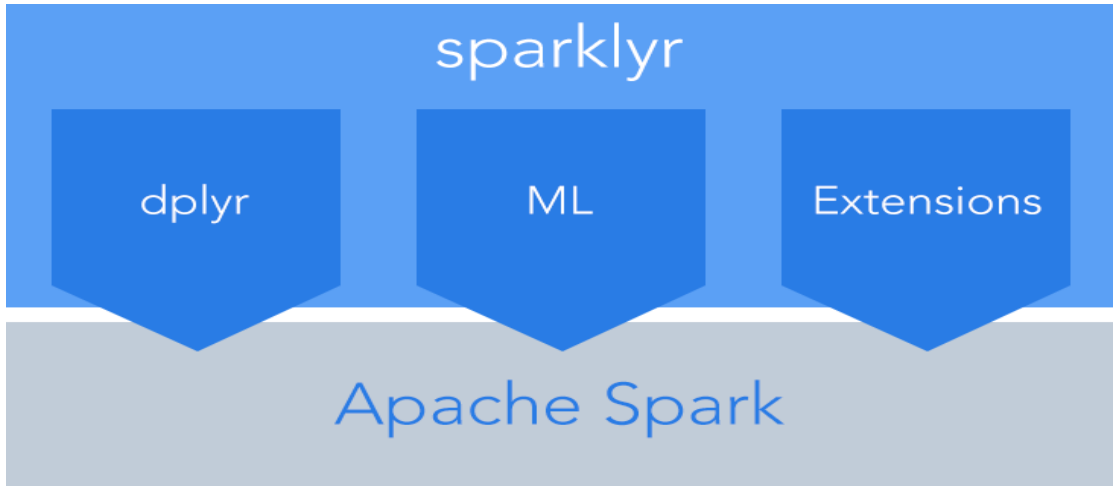
- Easy installation via devtools

```
# install sparklyr
devtools::install_github("rstudio/sparklyr")
```

- Connects to both local instance of Spark and to a Spark cluster

```
library(sparklyr)
config <- spark_config()
#connect to local instance of Spark
sc <- spark_connect(master = "local", config)
#connect to a Spark cluster
sc <- spark_connect(master = "spark://master_node_ip:7077",config)
```

sparklyr: R interface for Apache Spark

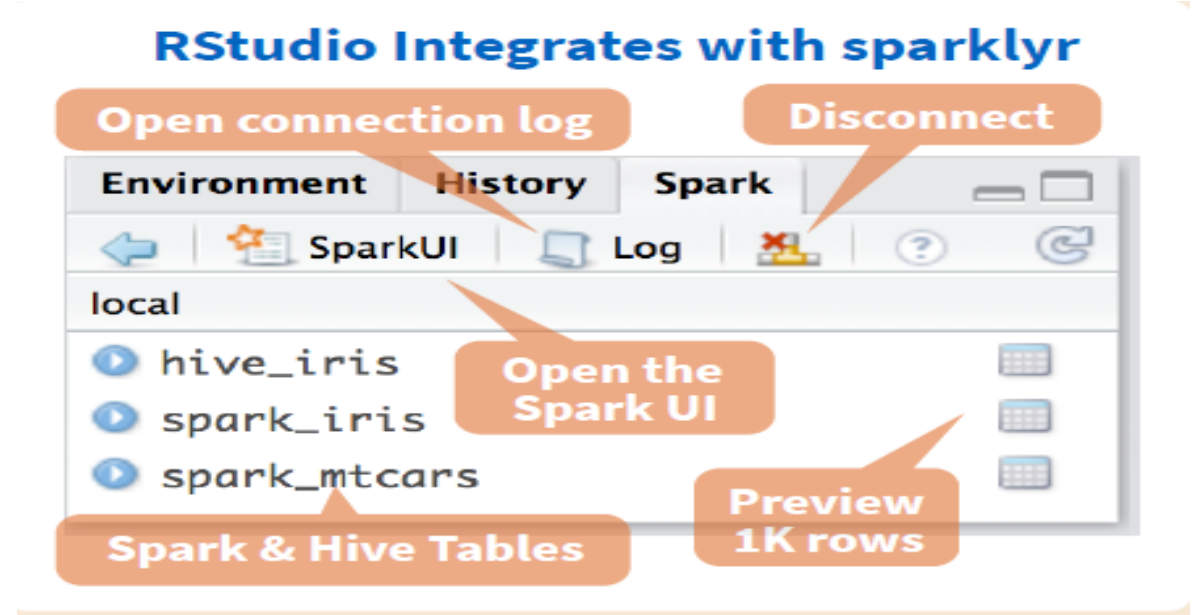


- Easy installation via devtools

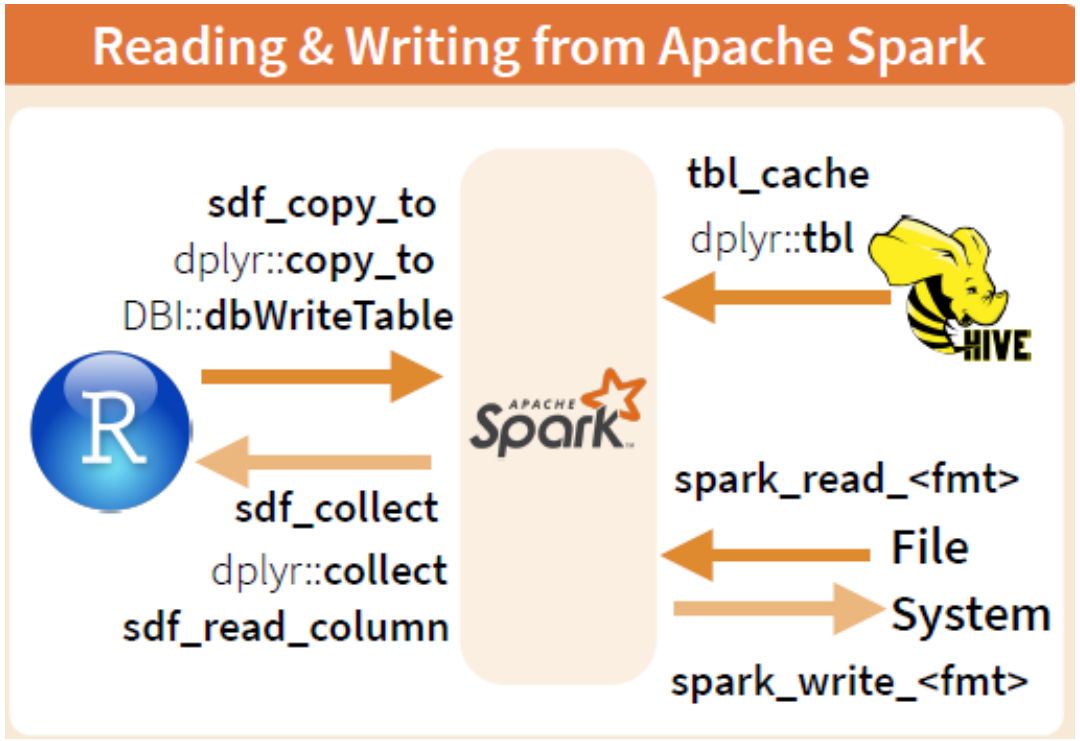
```
# install sparklyr
devtools::install_github("rstudio/sparklyr")
```

- Connects to both local instance of Spark and to a Spark cluster

```
library(sparklyr)
config <- spark_config()
#connect to local instance of Spark
sc <- spark_connect(master = "local", config)
#connect to a Spark cluster
sc <- spark_connect(master = "spark://master_node_ip:7077",config)
```



sparklyr: R interface for Apache Spark



- Loads data into Spark DataFrames from:
- Local R data frames (Small Data Set)
- Hive tables (Large Data Sets)
- Types: CSV, JSON, and Parquet files.

dplyr and ML in sparklyr

- Provides a complete dplyr backend for data manipulation, analysis and visualization

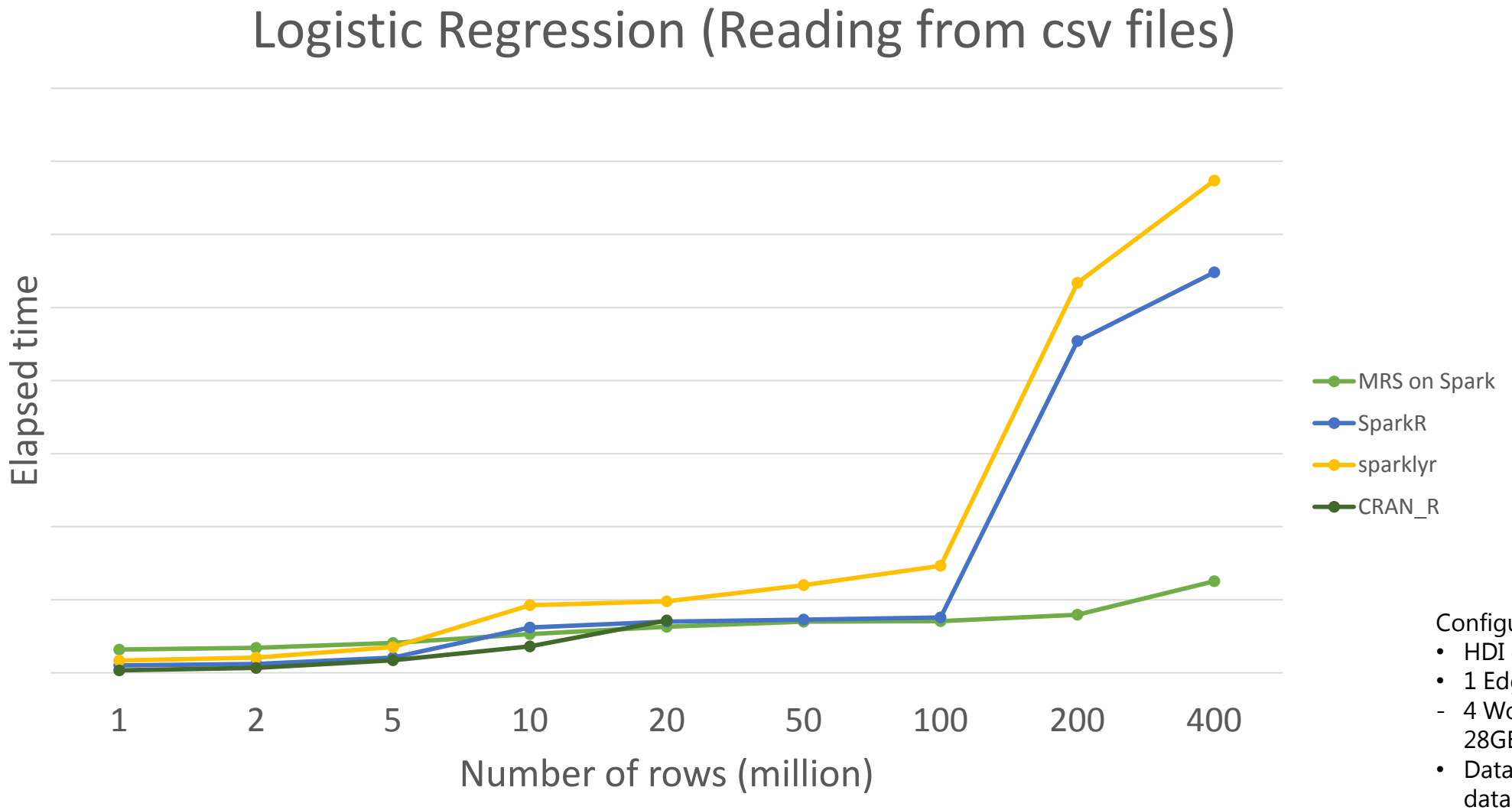
```
#manipulate data with dplyr
library(dplyr)
partitions <- data_spark %>%
  sdf_partition(training = 0.7, test = 0.3, seed = 1099)
```

%>%

- Includes 3 family of functions for machine learning
  - **ml_***: Machine learning algorithms for analyzing data provided by the *spark.ml* package.
  - **ft_***: Feature transformers for manipulating individual features.
  - **sdf_***: Functions for manipulating *SparkDataFrames*.

```
#Train RandomForest Model
rf_model <- partitions$training %>%
  ml_random_forest(response = class_name, features = features_names,
                   type = "classification", num.trees = 300, max.depth = 3)
```

R on Spark - faster and more scalable

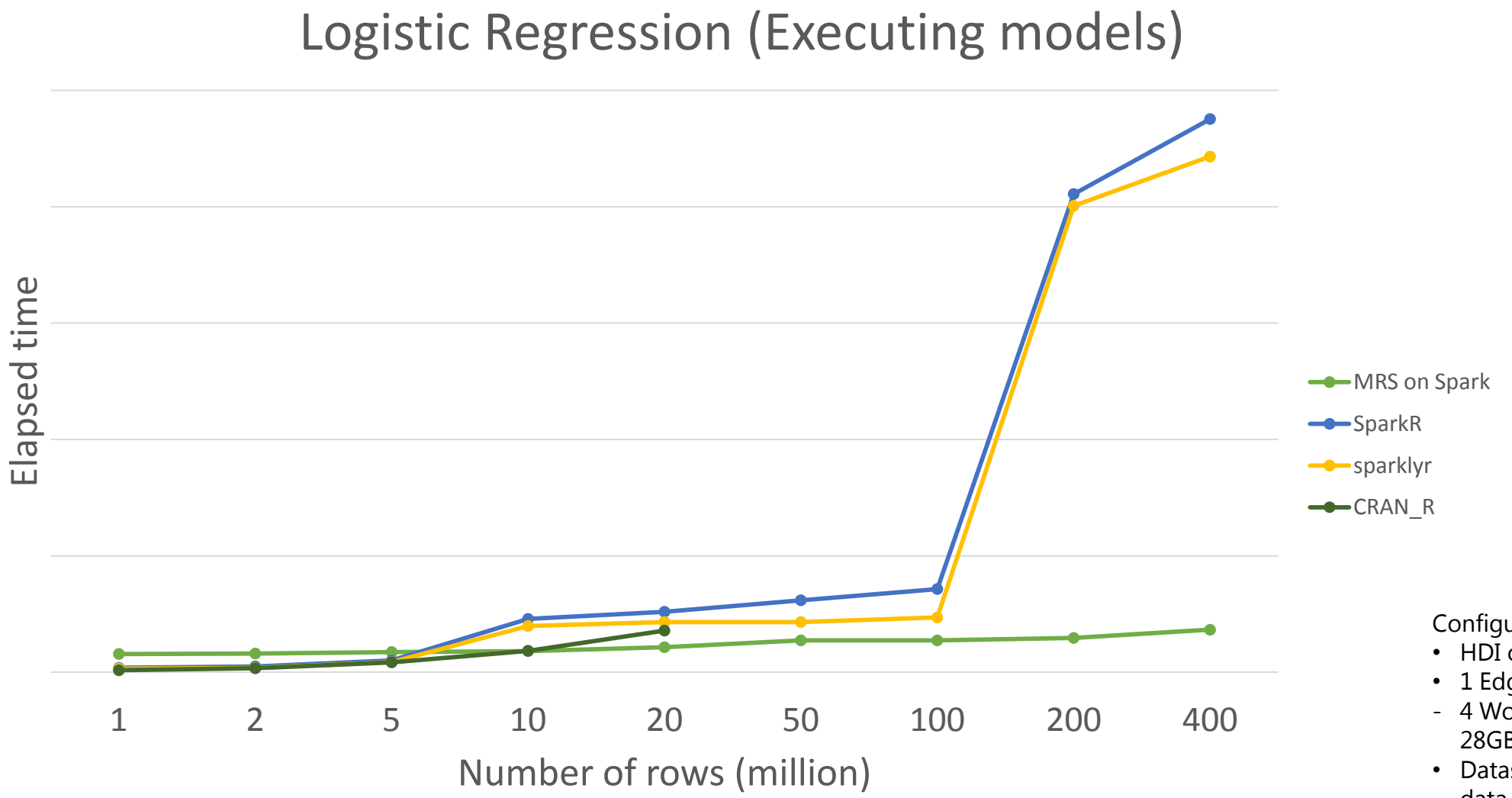


Configuration:

- HDI cluster size: 7 nodes
- 1 Edge Node: 8 cores, 28GB
- 4 Worker Nodes: 8 cores, 28GB
- Dataset: Duplicated Airlines data (.csv)
- Number of columns: 26

Source : Microsoft

R on Spark - substantially faster



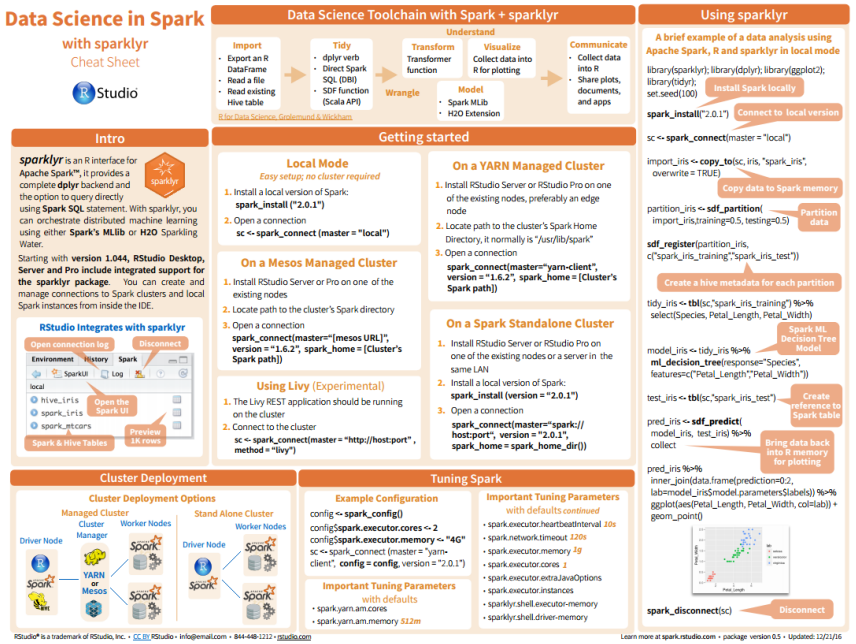
Configuration:

- HDI cluster size: 7 nodes
- 1 Edge Node: 8 cores, 28GB
- 4 Worker Nodes: 8 cores, 28GB
- Dataset: Duplicated Airlines data (.csv)
- Number of columns: 26

Source : Microsoft

Sparklyr cheat sheet

http://spark.rstudio.com/images/sparklyr-cheatsheet.pdf



Face recognition Model

Computer vision and its fields of application

Face recognition: a subfield of computer vision

Computer vision fields of application:

- Health: medical imaging
- Precision Agriculture: usage of computer vision for yields maximisation
- Optical Character Recognition (OCR): digitalization of handwritten documents
- Security: smart video surveillance: Intruder recognition

Face Recognition Architecture

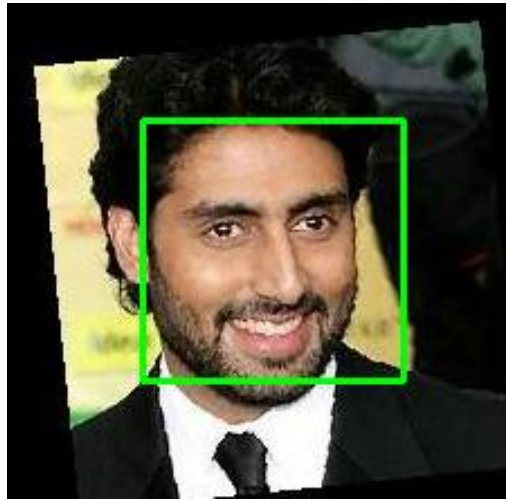
1. Generic Architecture

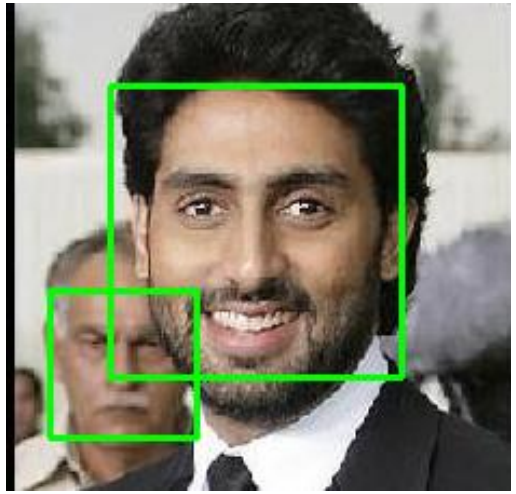
Face Recognition Architecture

2. Face Detection

Several techniques in the literature

- But, usually based on a supervised classifier model
- In our case: we used the OpenCV frontal-face model based on Haar features
- Training phase: we dropped the observations with 2 faces





Face Recognition Architecture

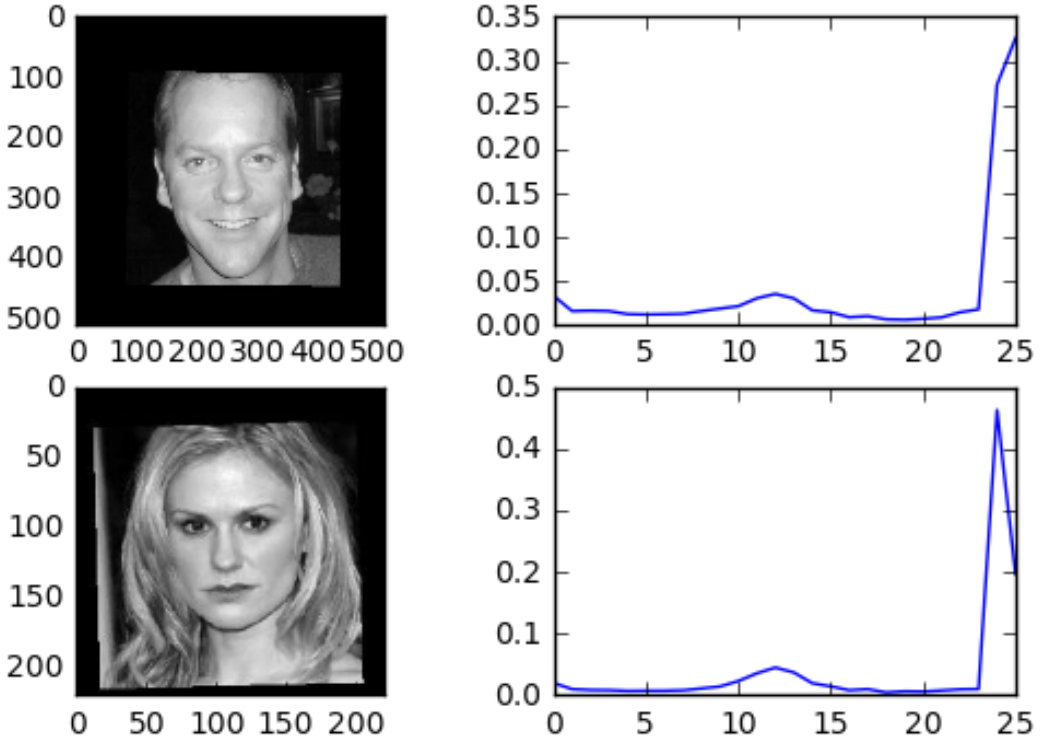
3. Features extraction (1/2)

- Goal: Extraction of discriminant information to feed a machine learning algorithm
- Features extraction ≈ Feature engineering applied to some attributes of the pixels
- Face recognition ➔ Local features are efficient
- Examples of Local features
  - HoG (Histogram of Oriented Gradient) ➔ Human image detection
  - Gabor filter ➔ Convolution of the image with Gabor kernels
  - Local Binary Pattern ➔ Useful for Texture description

Face Recognition Architecture

3. Features extraction (2/2) (LPB Histogram Example)

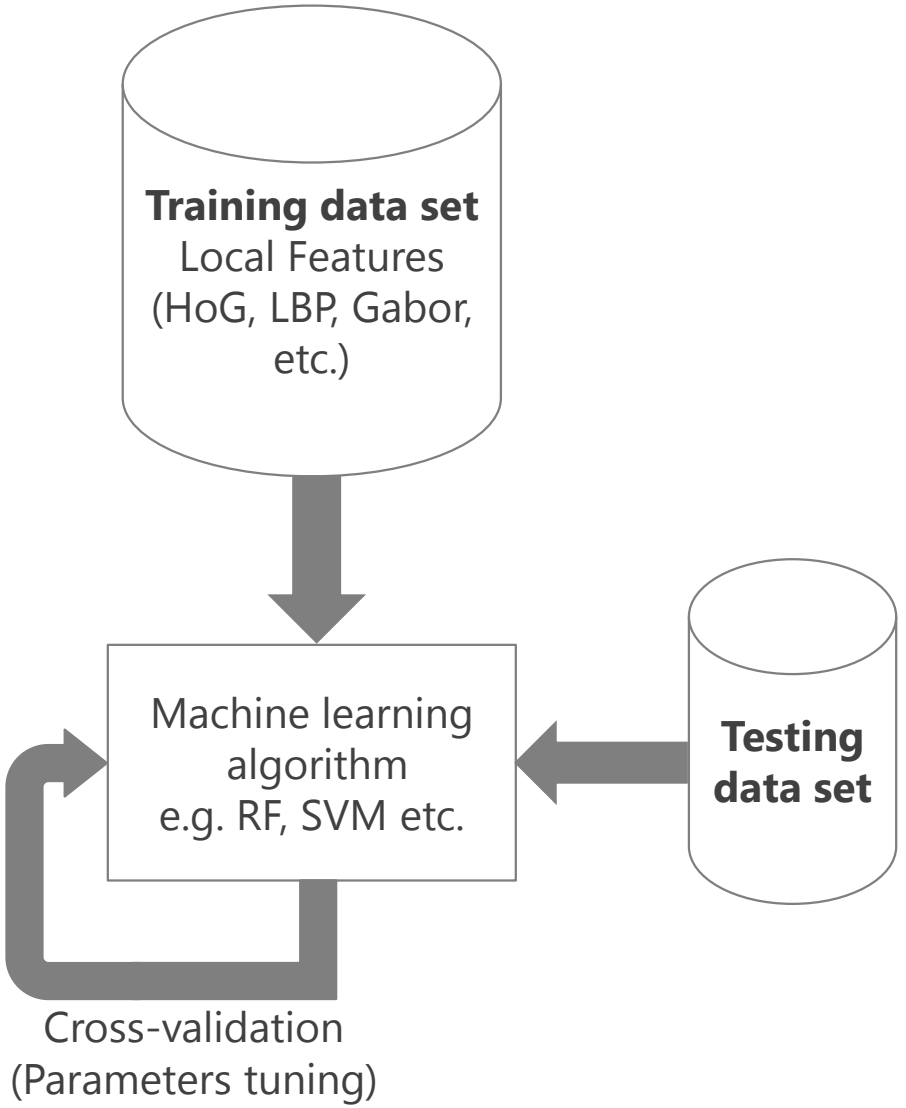
- LBP assumes: Face ≈ Set of micro-patterns
- We used histograms of LBP as features:
  ⇨ Each observation: 26 features (variables)



Face Recognition Architecture

4. Supervised Learning

- Feed a machine learning algorithm with features extracted:
  - SVM, Random Forests, Deep learning etc.
- Training/optimize a model using cross-validation process
- Test the model



Face Recognition Experimentation

1. Training the model

- Train and test data merged ⇨ global data set

- Training phase ⇨ split the global data set:
  - 70% for the train set and 30% for the test set

- Choice of machine learning algorithm:
  - ⇨Non parametric, Scalability and straightforward parameter tuning
  - ⇨Random forest (RF)

- We used SparklyR package to run Spark RF function

Face Recognition Experimentation

2. Testing the model

- Unfortunately: useless model obtained
- Probable reasons:
  - Inefficiency of the LBP features extracted (most important)
  - Lack of parameters tuning
- Need more times to investigate on the features extraction part:
  - A crucial component of the workflow
  - Combine several features ?
  - Automated features extraction using a deep learning algorithm ?

Conclusion

Conclusion

- Hadoop cluster : successfuly achieved
- Designed a face recognition engine : not really satisfactory
- Perspectives :

  - Combination of several features
  - Use a deep learning algorithm

THANK YOU

Q&A