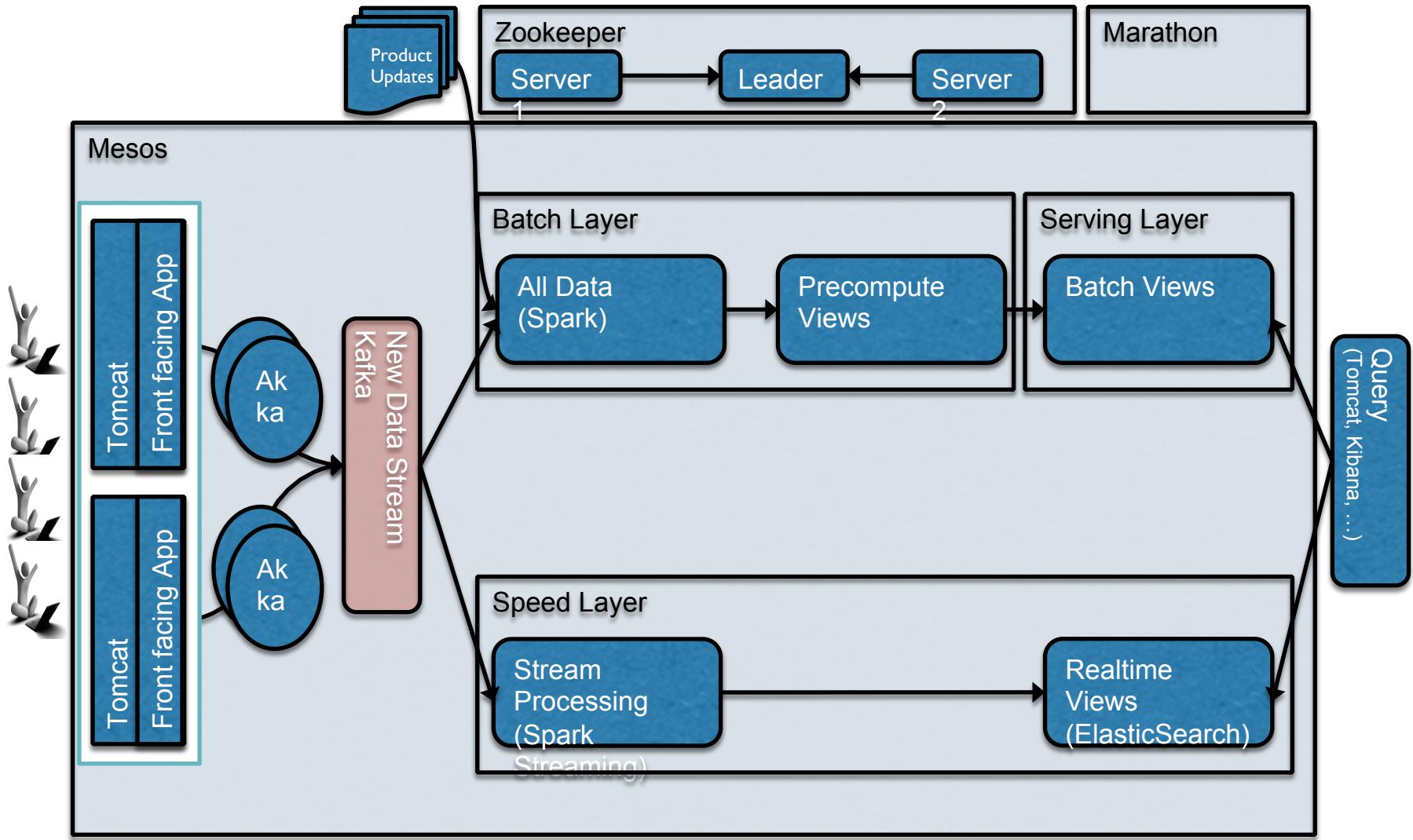


Applied Machine Learning with Apache Spark, MLlib and ElasticSearch

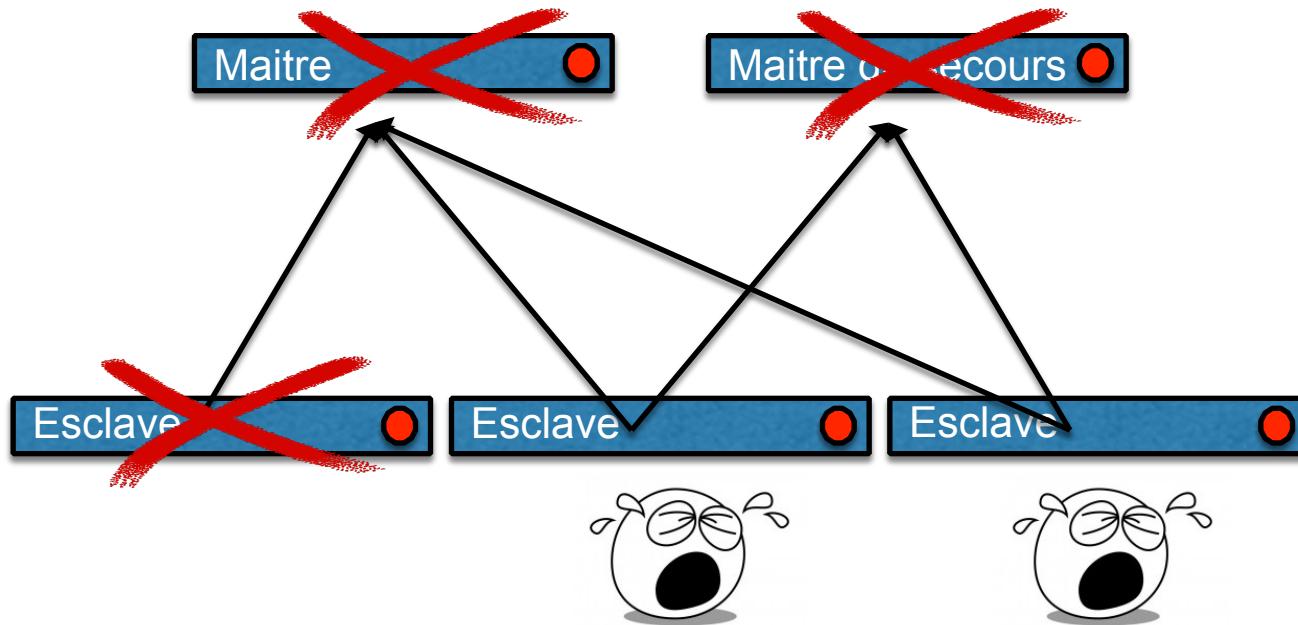
Hayssam Saleh – ebiznext

Cible



Zookeeper

Le problème



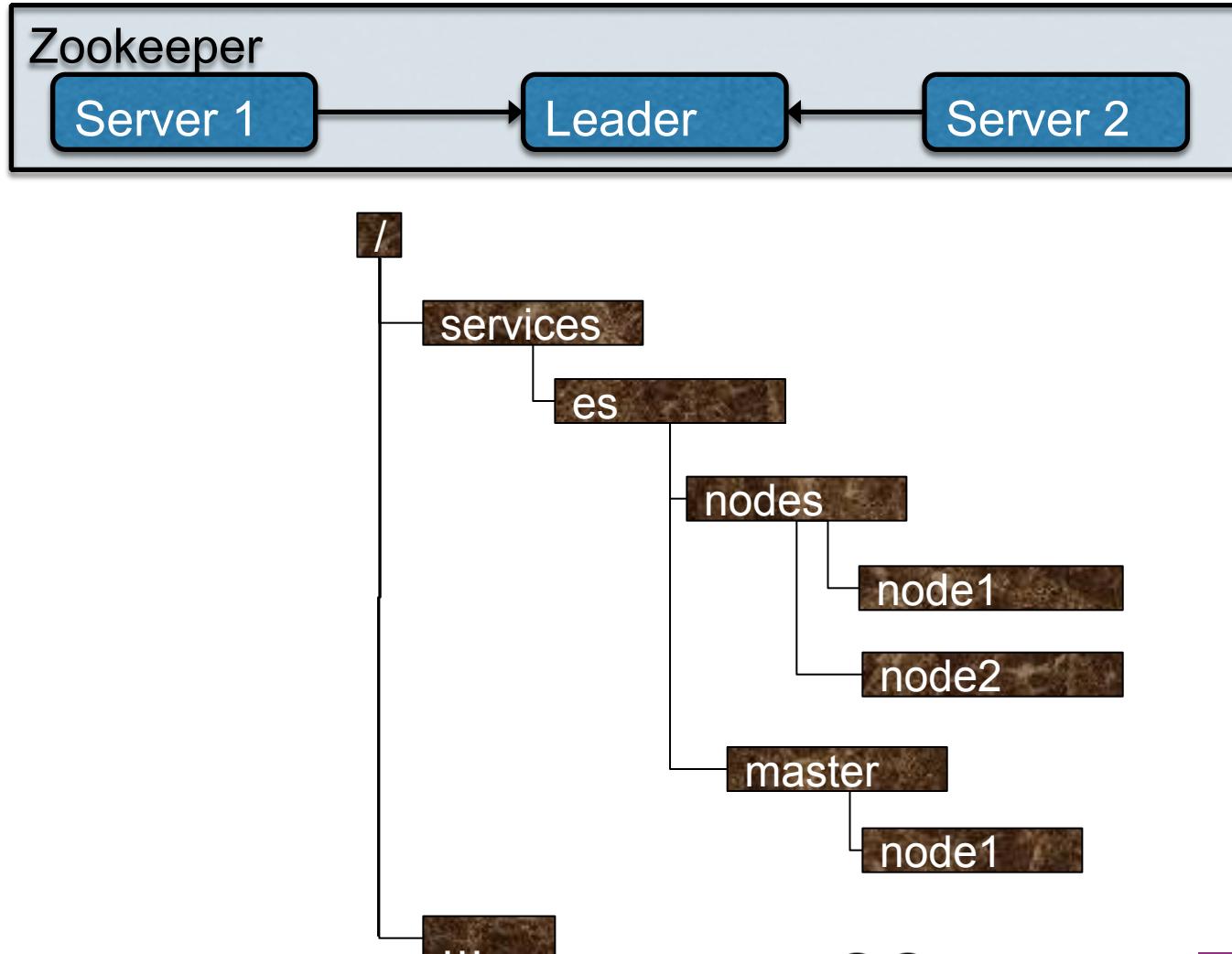
Zookeeper

Election de leader pour la tolérance au pannes

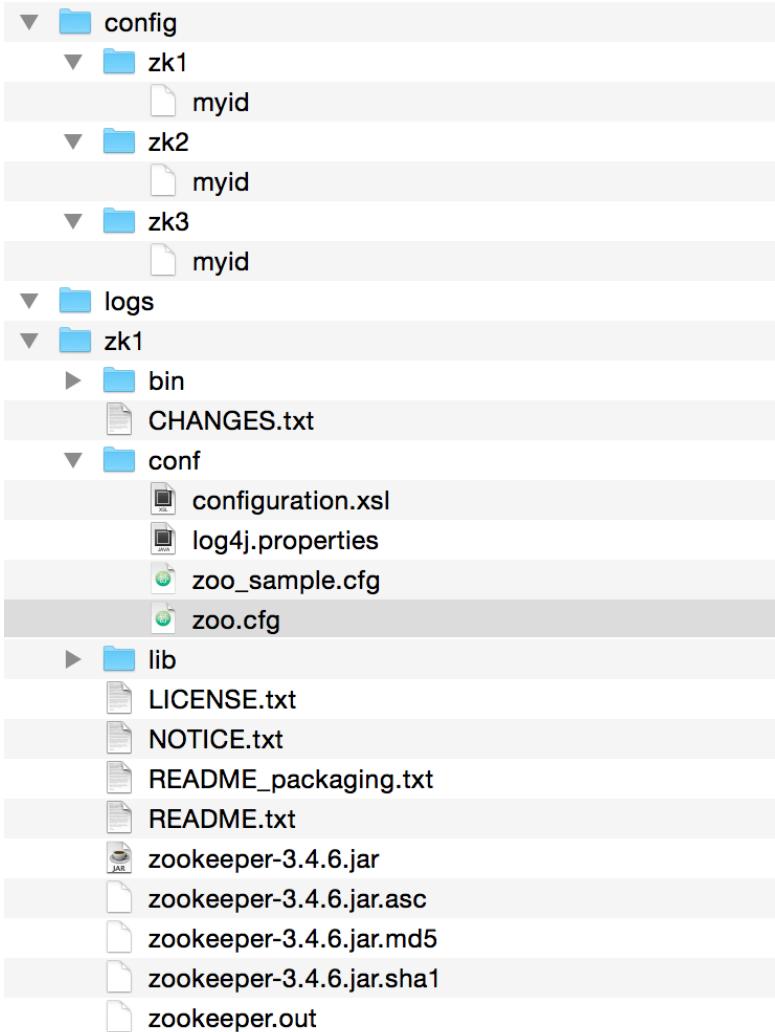


Avec Zookeeper
Tout noeud est un master potentiel

Arborescence Zookeeper



Installation Zookeeper



- Identification des instances
- Ports : Client / Qorum / Election
- Répertoire de données
- Répertoire de logs

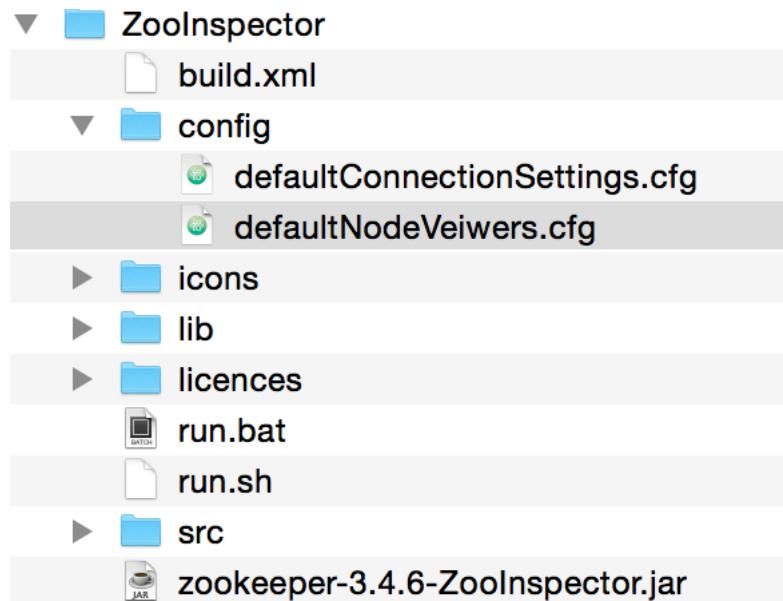


```
HAYSSAM:zk1 hayssams$ bin/zkServer.sh start-foreground
```

```
HAYSSAM:zk3 hayssams$ bin/zkServer.sh start-foreground
```

```
HAYSSAM:zk2 hayssams$ bin/zkServer.sh start-foreground
```

ZooInspector



```
#Default connection for ZooInspector  
hosts=localhost\:19901,localhost\:19902,localhost\:19903
```

```
java -cp zookeeper-3.4.6-ZooInspector.jar:lib/*:..../zk1/*:..../zk1/lib/* org.apache.zookeeper.inspector.ZooInspector
```

Zoolnspector

Connection Settings

Connect String	localhost:19901,localhost:19902,localhost:19903
Session Timeout	5000
Data Encryption Manager	org.apache.zookeeper.inspector.encryption.BasicDataEncryptionManager
Authentication Scheme	
Authentication Data	

Load from file **Set As Default** **OK** **Cancel**

ElasticSearch

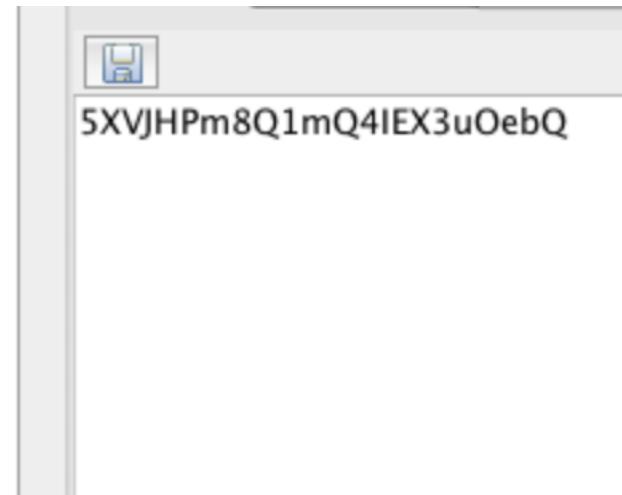
```
elasticsearch-1
  bin
  config
    elasticsearch.yml
    logging.yml
  data
  lib
    LICENSE.txt
  logs
    NOTICE.txt
  plugins
    head
    zookeeper
    README.textile
elasticsearch-2
```

```
cluster.name: scalaiocluster
index.number_of_shards: 1
index.number_of_replicas: 0
network.host: 127.0.0.1
transport.tcp.port: 19300
http.port: 19200
discovery.zen.ping.multicast.enabled: false
```

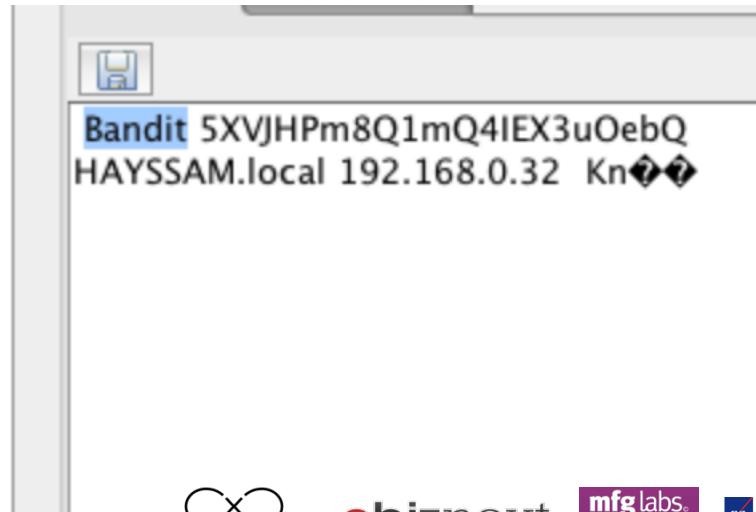
```
discovery:
  type: com.sonian.elasticsearch.zookeeper.discovery.ZooKeeperDiscoveryModule
sonian.elasticsearch.zookeeper:
  settings.enabled: true
  client.host: localhost:19901,localhost:19902,localhost:19903
  discovery.state_publishing.enabled: true
```

Zoolnspactor

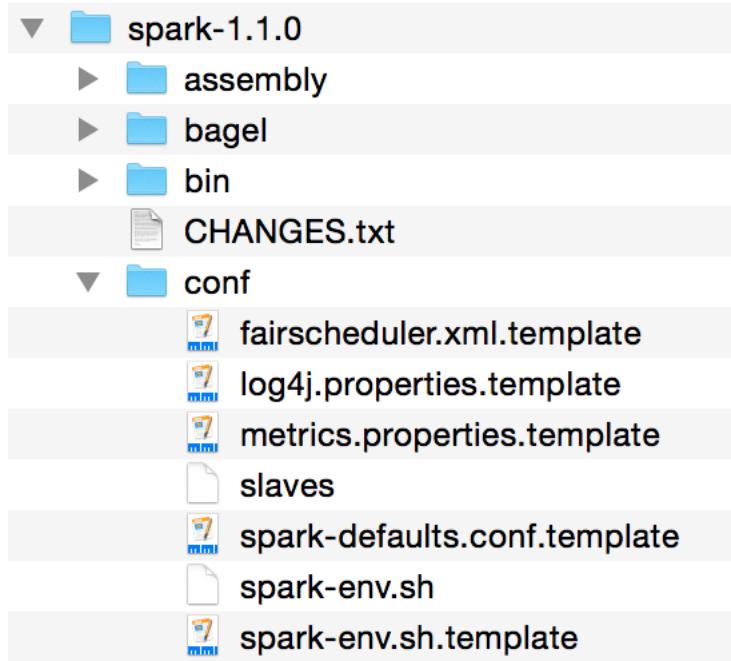
```
▼ es
  ▼ clusters
    ▼ scalaiocluster
      ▶ state
      ▼ nodes
        □ SXVJHPm8Q1mQ4IEX3uOebQ
        □ EQC8GWiHTDuDiOIGo78paQ
        □ leader
    ▶ zookeeper
```



```
▼ es
  ▼ clusters
    ▼ scalaiocluster
      ▶ state
      ▼ nodes
        □ SXVJHPm8Q1mQ4IEX3uOebQ
        □ EQC8GWiHTDuDiOIGo78paQ
        □ leader
    ▶ zookeeper
```

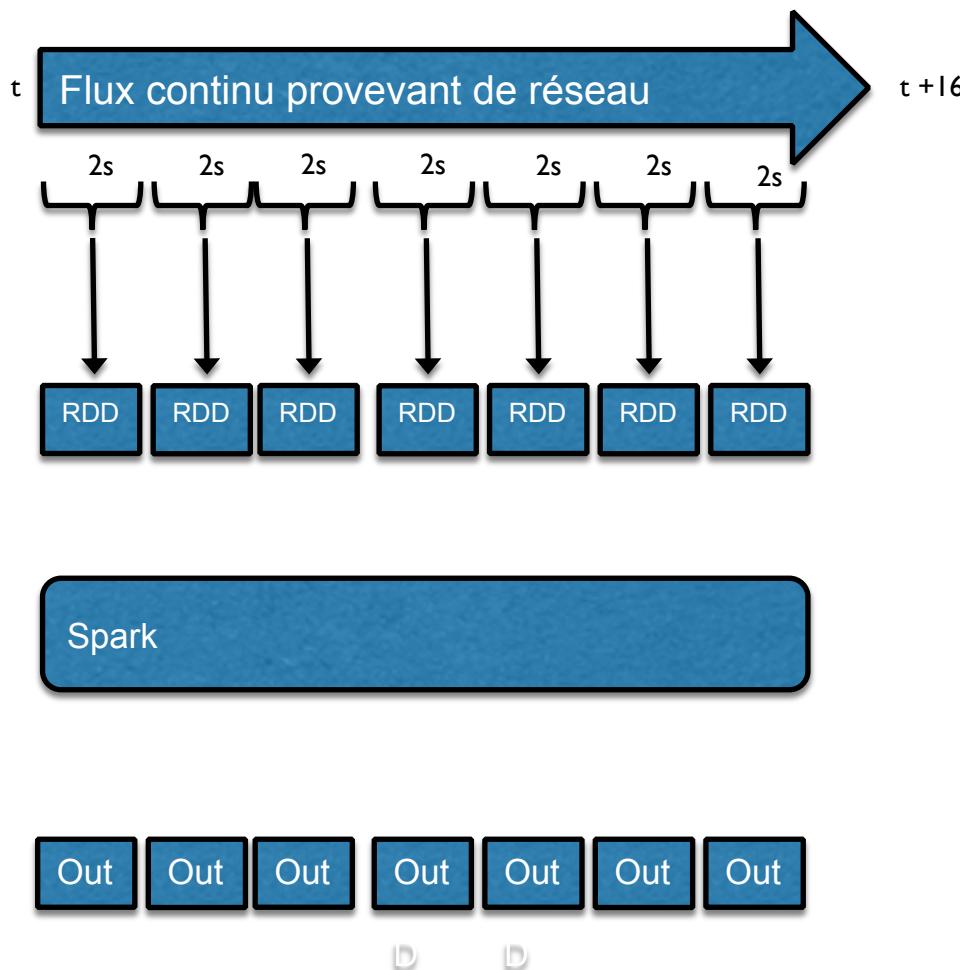


Spark



```
SPARK_LOCAL_IP=127.0.0.1  
SPARK_MASTER_IP=127.0.0.1  
SPARK_MASTER_PORT=19400  
SPARK_MASTER_WEBUI_PORT=19500  
SPARK_WORKER_WEBUI_PORT=19501
```

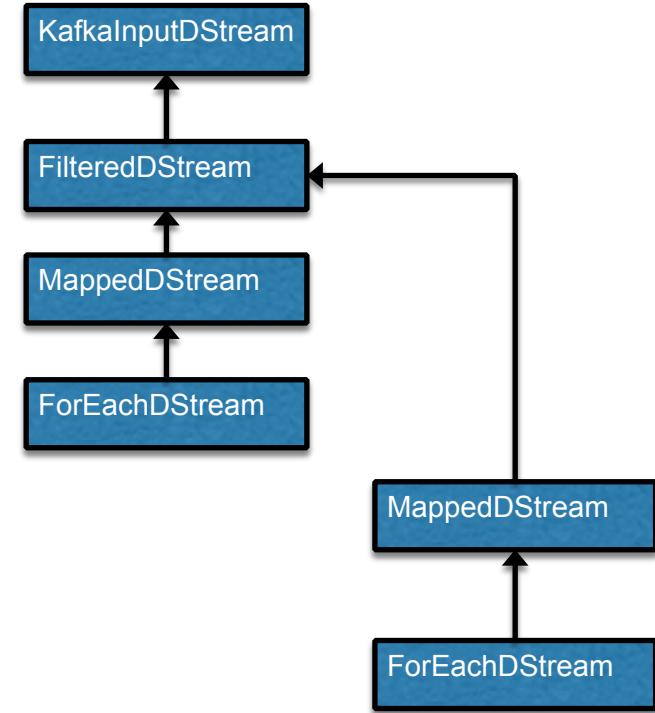
Avec Spark: API unique pour la batch et le streaming



- Découpage du flux en un paquet de données (RDD) toutes les 2 secondes
- Un RDD est une collection de données
- Chaque paquet est traité comme un RDD soumis au batch Spark classique
- Exécution de microbatchs
- Spark exécute les opérations sur le RDD comme dans un batch classique et renvoie le résultat en retour.

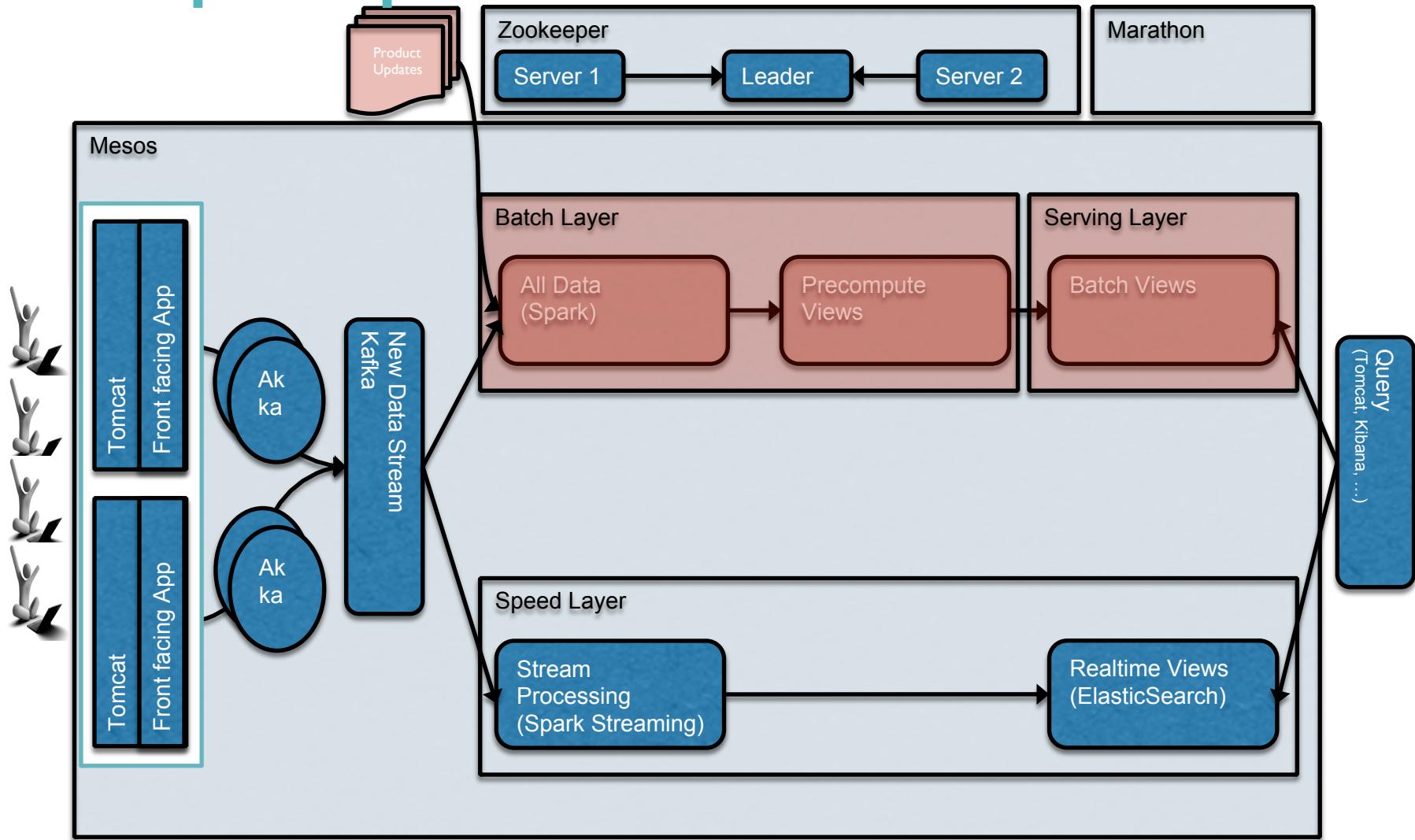
Avec Spark: Tolérance aux pannes au coût minimum

```
val tuples = kafka.initStream(context,"requests », 2  
seconds  
  
rdd = tuples.filter( t => RulesHandler.match(t))  
  
rdd2 = rdd.map( t => toDocument)  
  
rdd2.foreachRDD(doc => es.insert doc) // via Kafka  
  
rdd3 = rdd.flatMap(_.toRelations)  
  
rdd3.foreachRDD(rel => neo4j.insert rel) // via Kafka
```



- Le code ci-dessus ne génère aucun calcul, juste un graphe d'objets
- Le Scheduler Spark va soumettre le graphe d'objets aux workers pour exécution
- ☺ En cas de panne, il suffit juste de soumettre à nouveau le graphe d'objets à un autre worker.
 - Pas besoin de réPLICATION des résultats ou d'upstream backup

Pourquoi Spark Classic



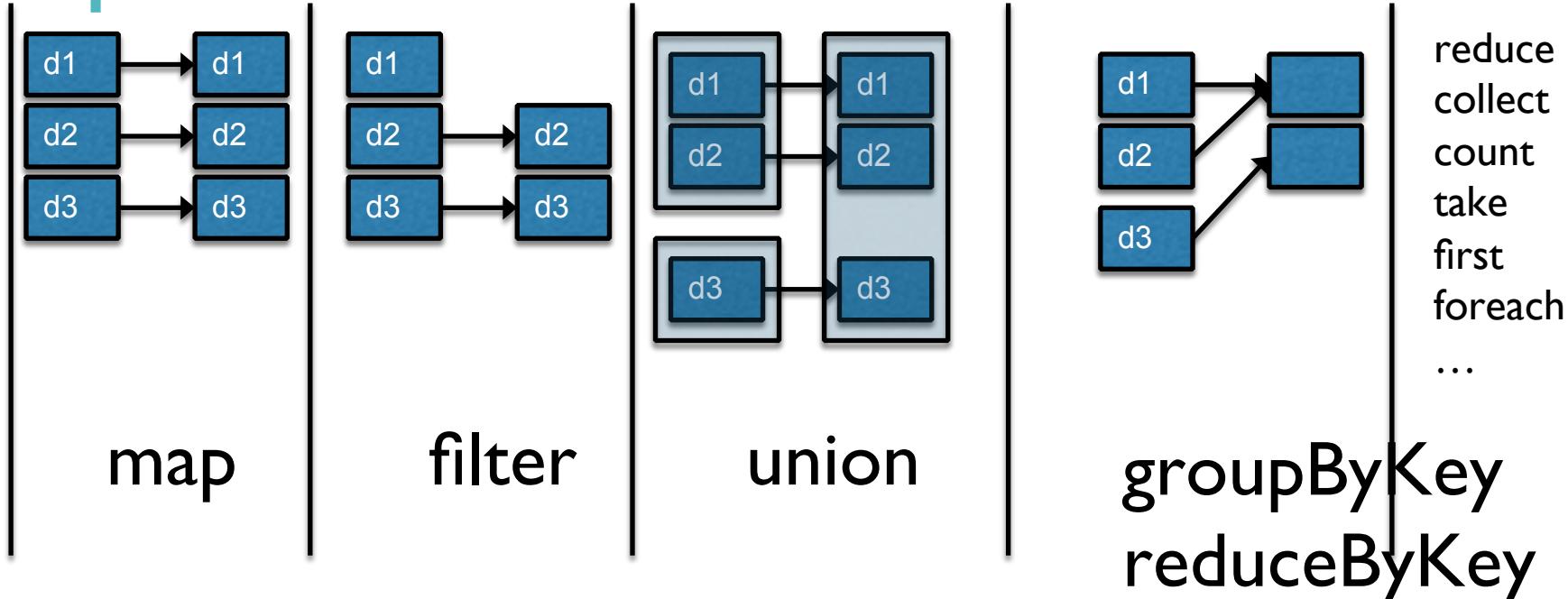
Notification des clients

RDD

```
Ordd= Sc.readFromXMLFile(...)  
Ordd.persist  
Oval updatedRDD = rdd.filter(product =>  
product.existsInMapWithDifferentHash)  
Oval newRDD = rdd.filter(product => !product.existInMap)  
  
Oval updateNotifications = updatedRDD.map(_.executeStrategies)  
  
Oval newNotifications = newRDD.map(_.executeStrategies)  
  
OupdateNotifications.union(newNotifications).foreachRDD(_.notif  
yClient)
```

Action

Avec Spark : Une riche librairie d'opérations



- Avec en plus
 - ☺ Contrôle sur le partitionnement
 - ☺ broadcast
 - ☺ accumulateurs
 - ☺ Les chaînage d'actions ne requiert pas d'écriture intermédiaire sur disque

Spark : Démarrage

```
SPARK_LOCAL_IP=127.0.0.1  
SPARK_MASTER_IP=127.0.0.1  
SPARK_MASTER_PORT=19400  
SPARK_MASTER_WEBUI_PORT=19500  
SPARK_WORKER_WEBUI_PORT=19501
```

```
HAYSSAM:spark-1.1.0 hayssams$ sbin/start-master.sh
```

```
HAYSSAM:spark-1.1.0 hayssams$ ./bin/spark-class org.apache.spark.deploy.worker.Worker spark://127.0.0.1:19400
```

Spark : Avec Zookeeper

```
export SPARK_DAEMON_JAVA_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER -Dspark.deploy.zookeeper.url=localhost:19901,localhost:19902,localhost:19903 -Dspark.deploy.zookeeper.dir=/spark"
```

```
HAYSSAM:spark-1.1.0 hayssams$ sbin/start-all.sh [
```

Analytics

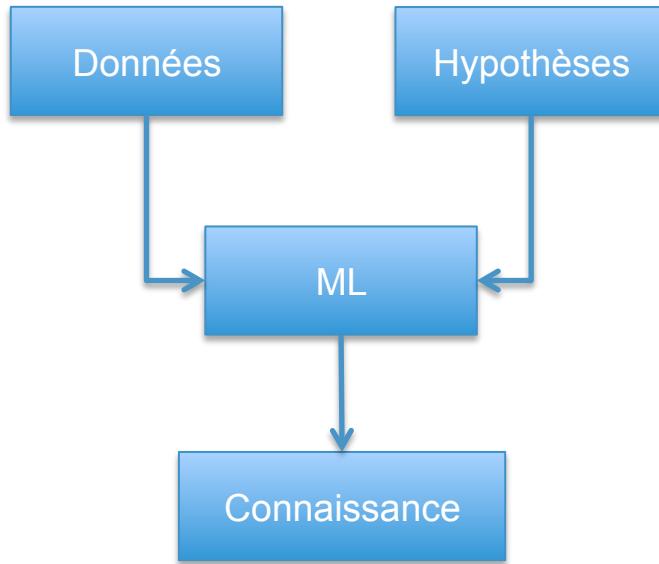
- La découverte de patterns dans les données
- S'appuie sur l'analyse et l'aggrégation de datasets conséquents
- Pour extraire des faits du passé
- Prédire des tendances
- En utilisant le machine learning et les outils de data mining
 - Machine Learning
 - Branche de l'intelligence artificielle
 - Capacité à apprendre sans être explicitement programmé
 - Exemple : Distinguer les mails de type spam des autres
 - Data mining
 - Processus de découverte de la connaissance
 - Découverte de patterns pour aider à la décision
 - Les outils de data mining utilisent les techniques de machine learning

Cas d'utilisation

- Recommandation
- Classification de contenu en groupes prédéfinis
- Regroupement de contenus similaires
- Recherche d'association/patterns dans les actions/comportements
- Identification de topics clefs
- Détection de fraude et d'anomalies
- Ranking

De la donnée à la connaissance

- De la donnée à
- la connaissance
 - Dois-je accorder ce crédit à ce client sur la base de ses bilans ?
 - Ce film est-il un film d'humour ?
 - Ce mail est-il un spam ?



Types d'apprentissage

● Apprentissage Supervisé

- Classification

- Le résultat est un valeur parmi N sans ordre quelconque et dans un ensemble prédéfini

- Exemple : Ce film est-il un film d'horreur ou un film d'actions ou un film romantique

- Prédire le label de nouvelles données à partir des labels des données existantes

- Régression

- Le résultat est une valeur dans un ensemble de valeurs continues

- Exemple : Quel est le prix prévisionnel de cet appartement dans les 6 prochains mois ?

● Apprentissage non supervisé

- Clustering

- Les données en entrée de l'algorithme d'apprentissage ne sont pas labellisées

- Grouper les données en fonction de leur similarité

- Exemple : ???

Apprentissage supervisé

- Prédire les labels de données futures à partir de labels de données existantes

Salaire	Statut	Propriétaire	OK/KO
100	Célibataire	Oui	OK
120	Marié	Non	OK
100	Divorcé	Non	KO
80	Célibataire	Non	KO
70	Marié	Non	KO
100	Célibataire	Oui	OK

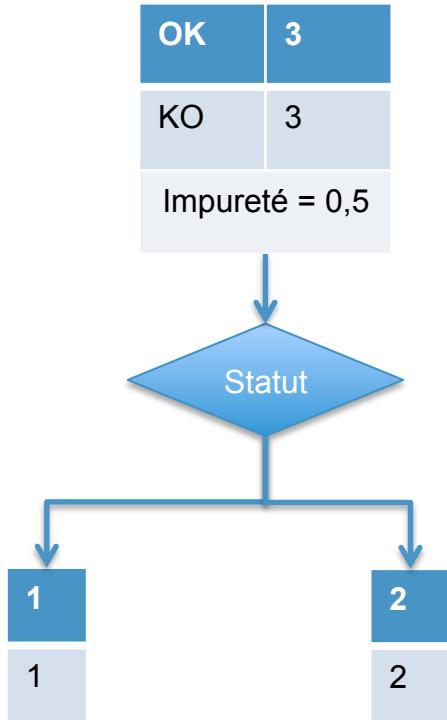
arbre de décision

$$\text{Entropy}(t) = - \sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t),$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2,$$

$$\text{Classification error}(t) = 1 - \max_i [p(i|t)],$$

Le meilleur arbre de décision

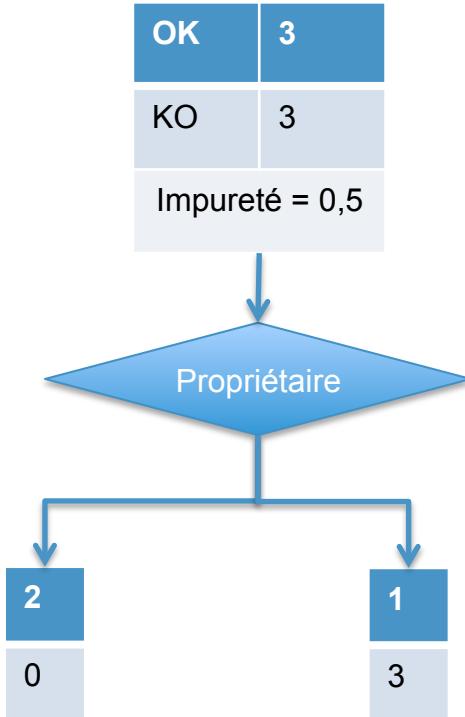


Statut ?	Marié	Célibataire / Divorcé
OK	1	2
KO	1	2

Gini (marié) = $1 - ((1/2)^2 + (1/2)^2) = 0,5$
Gini(Cél/Div) = $1 - ((2/2)^2 + (2/2)^2) = 0,5$
Impureté = $((1+1)/6) * \text{Gini(marié)} + ((2+2)/6) * \text{Gini(Cél/Div)} = 0,5$

Le meilleur arbre de décision

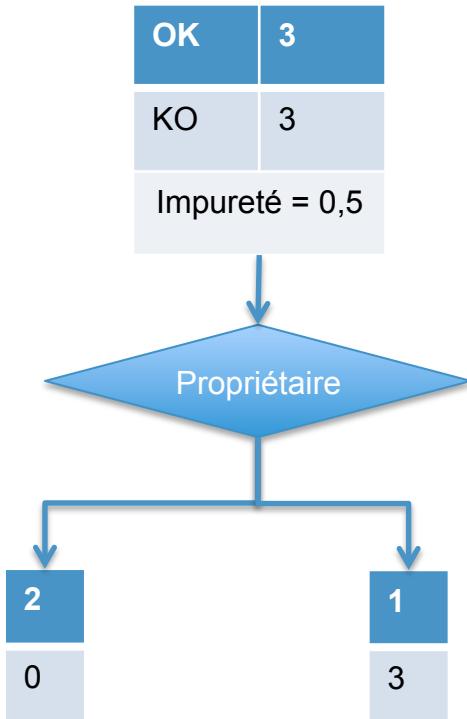
Utilisation de l'index Gini



Propriétaire ?	Oui	Non
OK	2	1
KO	0	3
$\text{Gini (Oui)} = 1 - ((2/2)^2 + (0/2)^2) = 0$ $\text{Gini(Non)} = 1 - ((1/4)^2 + (3/4)^2) = 0,375$ $\text{Impureté} = ((2+0)/6) * \text{Gini(Oui)} + ((1+3)/6) * \text{Gini(Non)} = 0,25$		
Marge d'erreur	$1 - \max(2/2, 0/2) = 0$	$1 - \max(1/4, 3/4) = 0,25$

Le meilleur arbre de décision

Utilisation de l'entropie



Propriétaire ?	Oui	Non
OK	2	1
KO	0	3
Entropie(Oui) = $-(2/2)\log_2(2/2) - (0/2)\log_2(0/2) = 0$		
Entropie(Non) = $-(1/4)\log_2(1/4) - (3/4)\log_2(3/4) = 0,5 + 0,31 = 0,81$		
Gain = $((2+0)/6) * \text{Entropie(Oui)} + ((1+3)/6) * \text{Entropie(Non)} = 0,4$		
Marge d'erreur	$1 - \max(2/2, 0/2) = 0$	$1 - \max(1/4, 3/4) = 0,25$

Arbre de décision

- Comment est construit l'arbre de décision
 - Recherche de la meilleure condition de segmentation
 - En s'appuyant sur l'impureté
 - Qu'est ce que la mesure d'impureté
 - Mesure la qualité de la séparation
- La segmentation s'arrête lorsque :
 - Tous les échantillons d'un nœud appartiennent à la même classe
 - Il n'y a plus de nouveaux attributs
 - Classification à la majorité
 - Il n'y a plus d'échantillons à classer
- Quelle technique Gini ou Entropie ?
 - La littérature :
 - Gini pour les attributs continus, l'entropie pour les catégories
 - L'entropie moins performante ?
 - Mon avis :
 - Essayer les deux et choisir celle qui fonctionne le mieux.

```

import org.apache.spark.SparkContext
import org.apache.spark.mllib.tree.DecisionTree
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.tree.configuration.Algo._
import org.apache.spark.mllib.tree.impurity.Gini

// Load and parse the data file
val data = sc.textFile("mllib/data/sample_tree_data.csv")
val parsedData = data.map { line =>
    val parts = line.split(',').map(_.toDouble)
    LabeledPoint(parts(0), Vectors.dense(parts.tail))
}

// Run training algorithm to build the model
val maxDepth = 5
val model = DecisionTree.train(parsedData, Classification, Gini, maxDepth)

// Evaluate model on training examples and compute training error
val labelAndPreds = parsedData.map { point =>
    val prediction = model.predict(point.features)
    (point.label, prediction)
}
val trainErr = labelAndPreds.filter(r => r._1 != r._2).count.toDouble / parsedData.count
println("Training Error = " + trainErr)

```

Classification naïve bayésienne

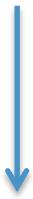
- Théorème de Bayes

- $p(y|x) = \frac{p(x|y) p(y)}{p(x)}$

- $p(y|x)$
 - Probabilité que x soit de la classe y
- $p(x|y)$
 - Quelle est dans l'échantillon labellisé, la probabilité que x soit dans la classe y
- $p(y)$
 - Quelle est dans l'échantillon labellisé, la probabilité d'occurrence de la classe y
- $p(x) =$
 - quelle est dans l'échantillon labellisé, la probabilité d'occurrence de x

Classification naïve bayésienne

$$P(\text{OK}|\text{Célibataire}) = \frac{p(\text{Célibataire}|\text{OK}) p(\text{OK})}{p(\text{Célibataire})}$$



$$P(\text{OK|oui}) = \frac{2/3 * 3/6}{3/6} = 0,67$$

Statut	OK/KO
Célibataire	OK
Marié	OK
Divorcé	KO
Célibataire	KO
Marié	KO
Célibataire	OK

Classification naïve bayésienne

Client	Salaire	Statut	Propriétaire	OK/KO
C1	100 = 2	Célibataire	Oui	OK
C2	120 = 2	Marié	Non	OK
C3	100 = 2	Divorcé	Non	KO
C4	80 = 1	Célibataire	Non	KO
C5	70 = 1	Marié	Non	KO
C6	100 = 2	Célibataire	Oui	OK
C7	90 = 1	Marié	Non	

$$p(x|y) = p(x_1|y) p(x_2|y) p(x_3|y)$$

$$p(C7|OK) = p(1|OK) p(Marié|OK) p(Non|OK) = 0 * 0.5 * 0.25$$

```

import org.apache.spark.mllib.classification.NaiveBayes
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint

val data = sc.textFile("mllib/data/sample_naive_bayes_data.txt")
val parsedData = data.map { line =>
    val parts = line.split(',')
    LabeledPoint(parts(0).toDouble, Vectors.dense(parts(1).split(' ').map(_.toDouble)))
}
// Split data into training (60%) and test (40%).
val splits = parsedData.randomSplit(Array(0.6, 0.4), seed = 11L)
val training = splits(0)
val test = splits(1)

val model = NaiveBayes.train(training, lambda = 1.0)
val prediction = model.predict(test.map(_.features))

val predictionAndLabel = prediction.zip(test.map(_.label))
val accuracy = 1.0 * predictionAndLabel.filter(x => x._1 == x._2).count() / test.count()

```

Régression logistique

$$\ln \left[\frac{P(y = 1 | x)}{P(y = 0 | x)} \right] = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3$$

$$\ln \left[\frac{\pi(x)}{1 - \pi(x)} \right] = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3$$

$$\pi(x) = \frac{e^{(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3)}}{1 - e^{(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3)}}$$

- Consiste à donner un poids à chaque prédicteur.
- Pour obtenir un rapport de chances.

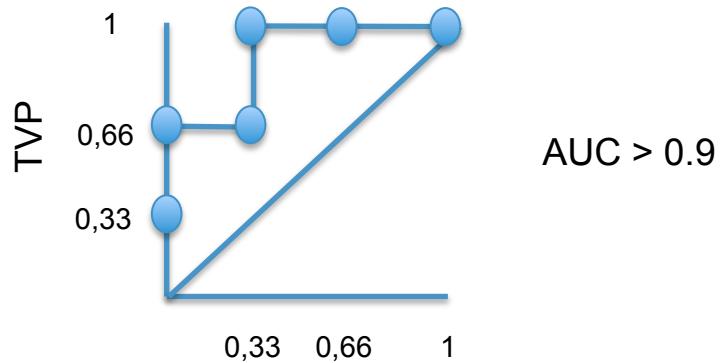
Régression logistique

Client	Salair e	Statut	Propriétaire	OK/KO	C(X)	PI (Score)
C1	100	Célibataire	Oui	OK	1,99	0,87
C2	120	Marié	Non	OK	1,3	0,55
C3	100	Divorcé	Non	KO	-2,7	0,12
C4	80	Célibataire	Non	KO	-1,5	0,01
C5	70	Marié	Non	KO	-0,38	0,42
C6	100	Célibataire	Oui	OK	-0,49	0,21

Régression logistique – La courbe ROC

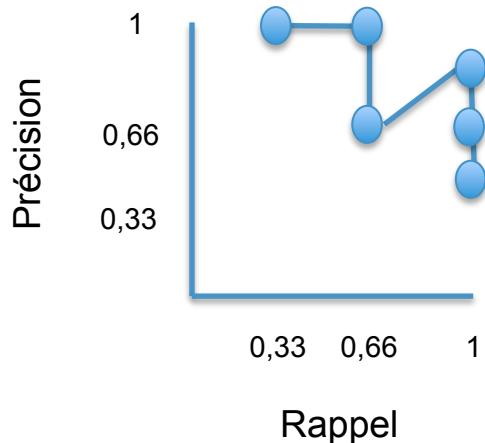
Client	Salair e	Statut	Proprié taire	OK/KO	C(X)	PI (Score)	TVP	TFP
C1	100	Célib.	Oui	OK	1,99	0,87	1/3 = 0,33	0/3 = 0
C2	120	Marié	Non	OK	1,3	0,55	2/3 = 0,66	0/3 = 0
C5	70	Marié	Non	KO	-0,38	0,42	2/3 = 0,66	1/3 = 0,33
C6	100	Célib.	Oui	OK	-0,49	0,21	3/3 = 1	1/3 = 0,33
C3	100	Divor cé	Non	KO	-2,7	0,12	3/3 = 1	2/3 = 0,66
C4	80	Célib.	Non	KO	-1,5	0,01	3/3 = 1	3/3 = 1

AUC	
AUC = 0.5	Pas de discrimination
0.7 < AUC < 0.8	Discrimination acceptable
0.8 < AUC < 0.9	Discrimination excellente
AUC >= 0.9	Discrimination excellente



Regression logistique – Courbe rappel - précision

Client	Salai re	Statut	Proprié taire	OK/KO	C(X)	PI (Score)	Rappel	Précision
C1	100	Célib.	Oui	OK	1,99	0,87	1/3 = 0,33	1/1 = 1
C2	120	Marié	Non	OK	1,3	0,55	2/3 = 0,66	2/2 = 1
C5	70	Marié	Non	KO	-0,38	0,42	2/3 = 0,66	2/3 = 0,66
C6	100	Célib.	Oui	OK	-0,49	0,21	3/3 = 1	3 / 4 = 0,75
C3	100	Divorc é	Non	KO	-2,7	0,12	3/3 = 1	3 / 5 = 0,6
C4	80	Célib.	Non	KO	-1,5	0,01	3/3 = 1	3 / 6 = 0,5



- PR versus ROC
 - PR plus adaptée lorsque les classes sont très déséquilibrées ?

```
import org.apache.spark.SparkContext
import org.apache.spark.mllib.classification.SVMWithSGD
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.util.MLUtils

// Load training data in LIBSVM format.
val data = MLUtils.loadLibSVMFile(sc, "mllib/data/sample_libsvm_data.txt")

// Split data into training (60%) and test (40%).
val splits = data.randomSplit(Array(0.6, 0.4), seed = 11L)
val training = splits(0).cache()
val test = splits(1)

// Run training algorithm to build the model
val numIterations = 100
val model = LogisticRegressionWithSGD.train(training, numIterations)

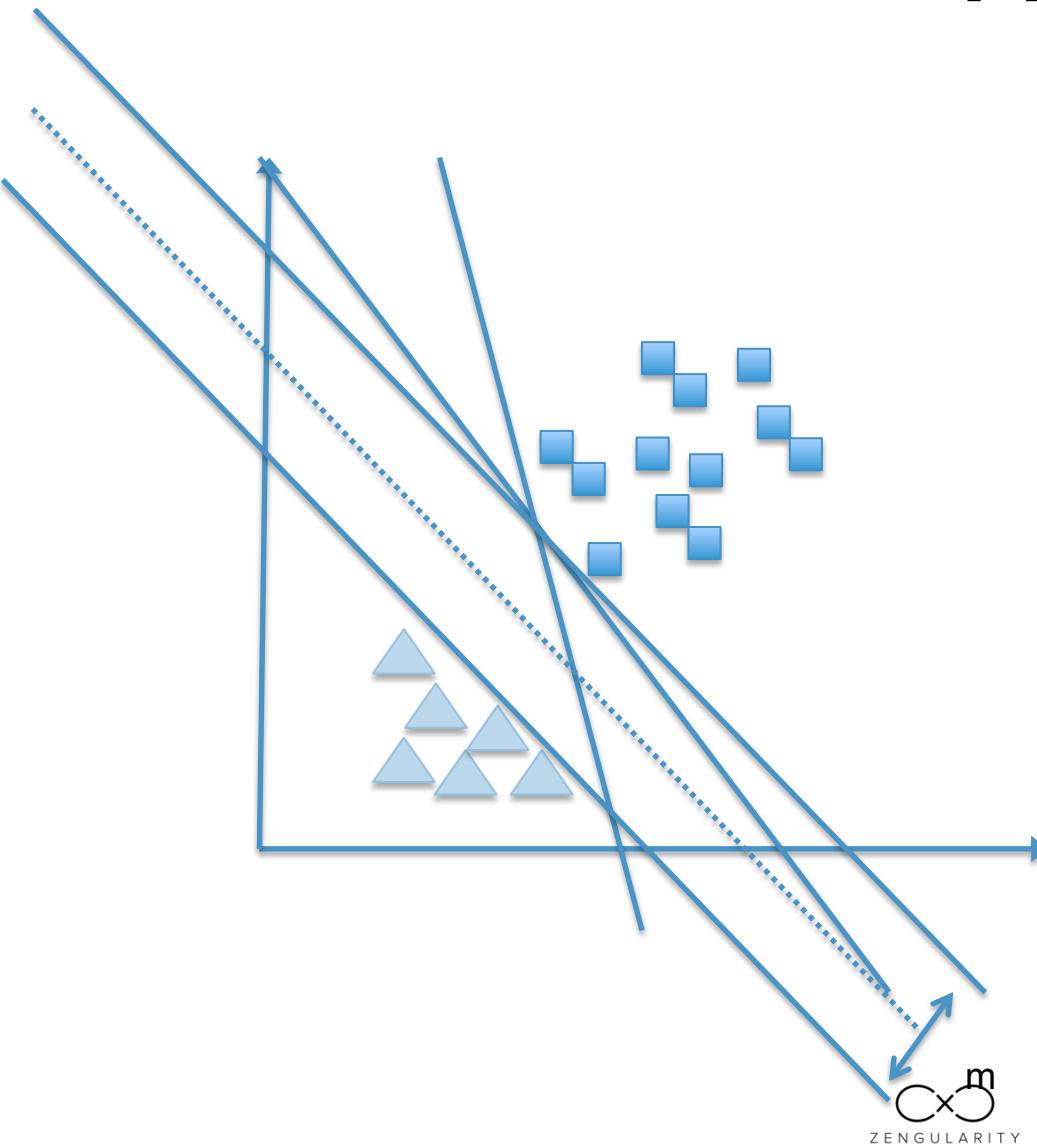
// Clear the default threshold.
model.clearThreshold()

// Compute raw scores on the test set.
val scoreAndLabels = test.map { point =>
    val score = model.predict(point.features)
    (score, point.label)
}

// Get evaluation metrics.
val metrics = new BinaryClassificationMetrics(scoreAndLabels)
val auROC = metrics.areaUnderROC()

println("Area under ROC = " + auROC)
```

Machine à vecteur de support



Régression linéaire

- Classification

- La classe est une variable discrète

- La régression

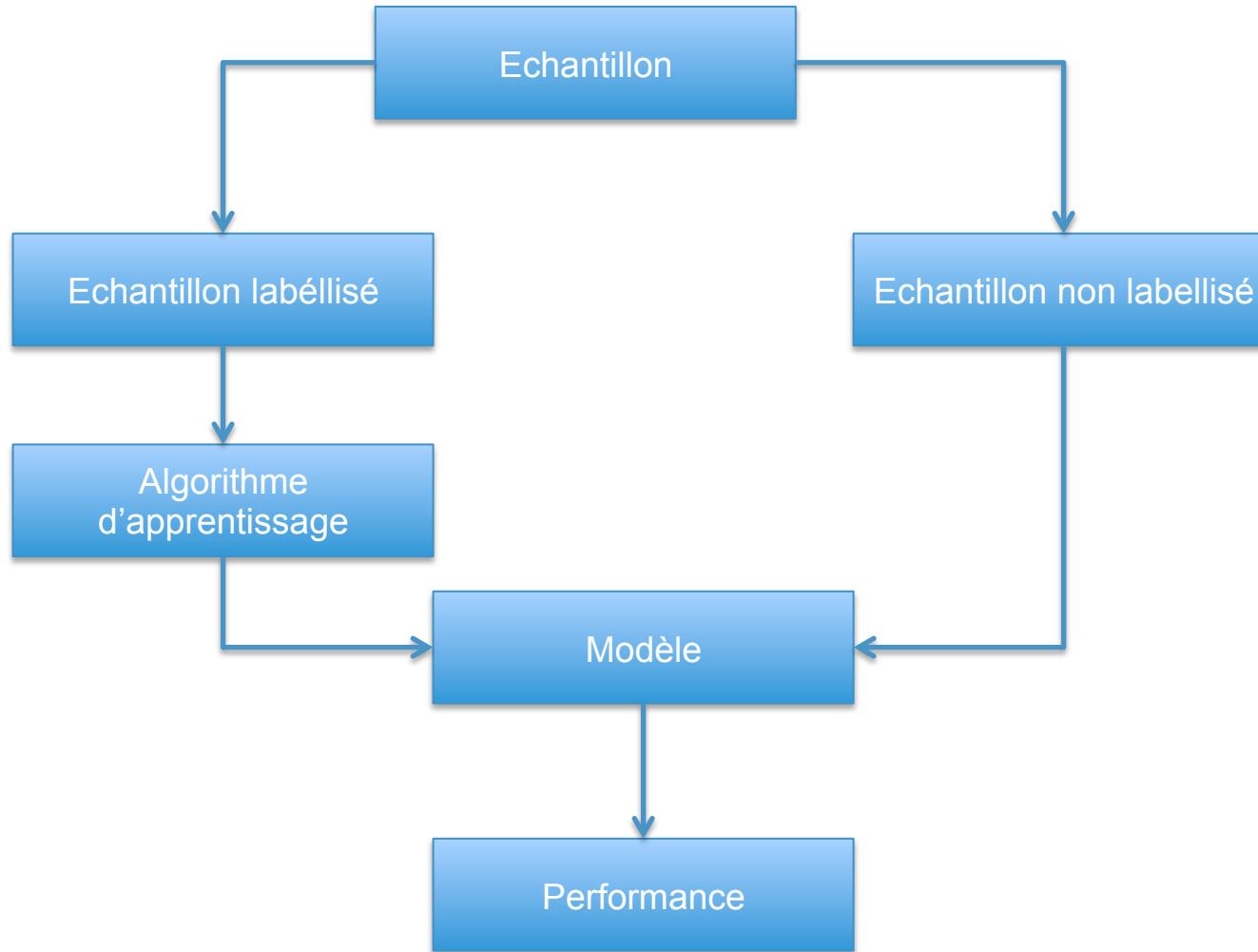
- La classe est une valeur continue

- Dans les deux cas nous sommes toujours dans l'apprentissage supervisé.

- Exemple

- Quelle sera la fréquentation du Stade de France lors de la prochaine coupe d'Europe

Démarche



- Le modèle génère les prédictions
- Le modèle est corrigé lorsque les prédictions sont erronées
- Le modèle est considéré satisfaisant lorsque le niveau de précision est considéré comme étant satisfaisant

Apprentissage Non supervisé

Clustering – plus proches voisins

- Les données en entrée ne sont pas labélisées
- Le clustering
 - va permettre de regrouper les données similaires en cluster

● Objectifs

Utilisateur	Age	Enfants	Sexe
U1	38	2	Homme
U2	25	0	Femme
U3	45	3	
U4	28	0	

Distance entre U1 et U2 = racine²((38-25)² + (2-0)²) = 13

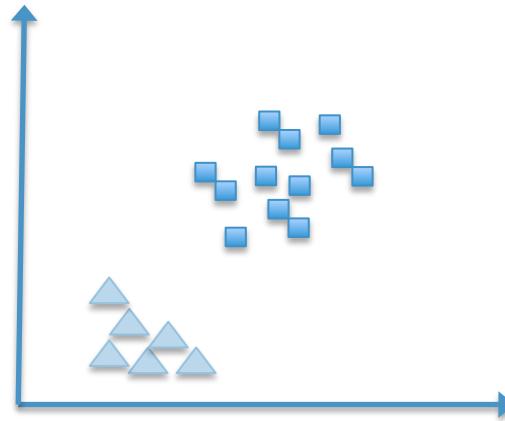
Distance entre U1 et U3 = 7

Clustering : Apprentissage non supervisé

- Objectif

- Calculer la distance entre les items en pondérant les valeurs des features
- Distance entre deux documents permet d'identifier des plagiats par exemple

Feature/Item	P1	P2	P3	P4
CPU	3.2	2.7	1.4	2.9
SSD	1	1	0	0
Disque	1To	512G	512G	1To
RAM	32Go	16Go	8Go	64G



```
import org.apache.spark.mllib.clustering.KMeans
import org.apache.spark.mllib.linalg.Vectors

// Load and parse the data
val data = sc.textFile("data/kmeans_data.txt")
val parsedData = data.map(s => Vectors.dense(s.split(' ').map(_.toDouble)))

// Cluster the data into two classes using KMeans
val numClusters = 2
val numIterations = 20
val clusters = KMeans.train(parsedData, numClusters, numIterations)

// Evaluate clustering by computing Within Set Sum of Squared Errors
val centers : Array[Array[Double]] = clusters.clusterCenters()
```

Collaborative filtering

Alternating Least Squares

	SW1	SW2	V1	V2
U1	4		4	
U2	5	5		
U3				1
U1000		4	1	

- Objectif

- Bâtir une matrice de rang inférieur qui permette de faire de la recommandation

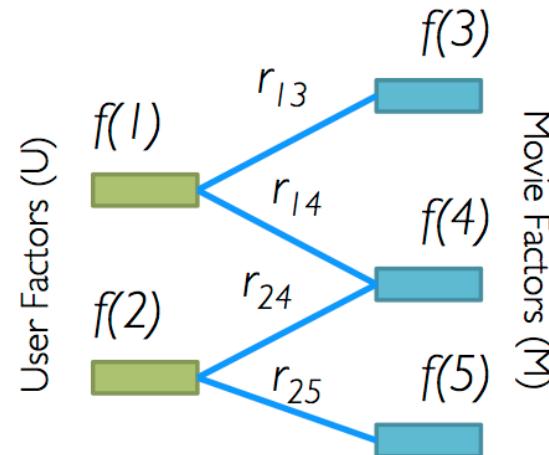
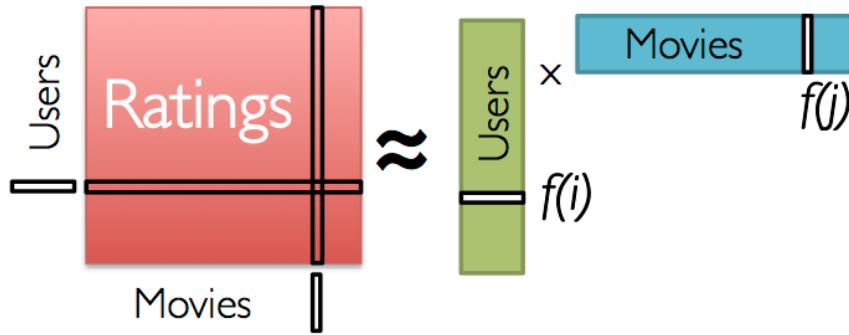
Collaborative filtering Alternating Least Squares

$$\mathbf{M} = \begin{pmatrix} 5 & 2 & 9 \\ 2 & 4 & 10 \\ 1 & 3 & 7 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 2 \\ 2 & 4 \end{pmatrix} \text{ or } \begin{pmatrix} 3 & 7 \\ 2 & 9 \end{pmatrix} \text{ or } \begin{pmatrix} 5 & 2 \\ 1 & 3 \end{pmatrix} \text{ etc.}$$

Collaborative Filtering ALS

Low-Rank Matrix Factorization:



Iterate:

$$f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \text{Nbrs}(i)} (r_{ij} - w^T f[j])^2 + \lambda ||w||_2^2$$

Référence : Amplab

```
import org.apache.spark.mllib.recommendation.ALS
import org.apache.spark.mllib.recommendation.Rating

// Load and parse the data
val data = sc.textFile("mllib/data/als/test.data")
val ratings = data.map(_.split(',') match { case Array(user, item, rate) =>
    Rating(user.toInt, item.toInt, rate.toDouble)
})

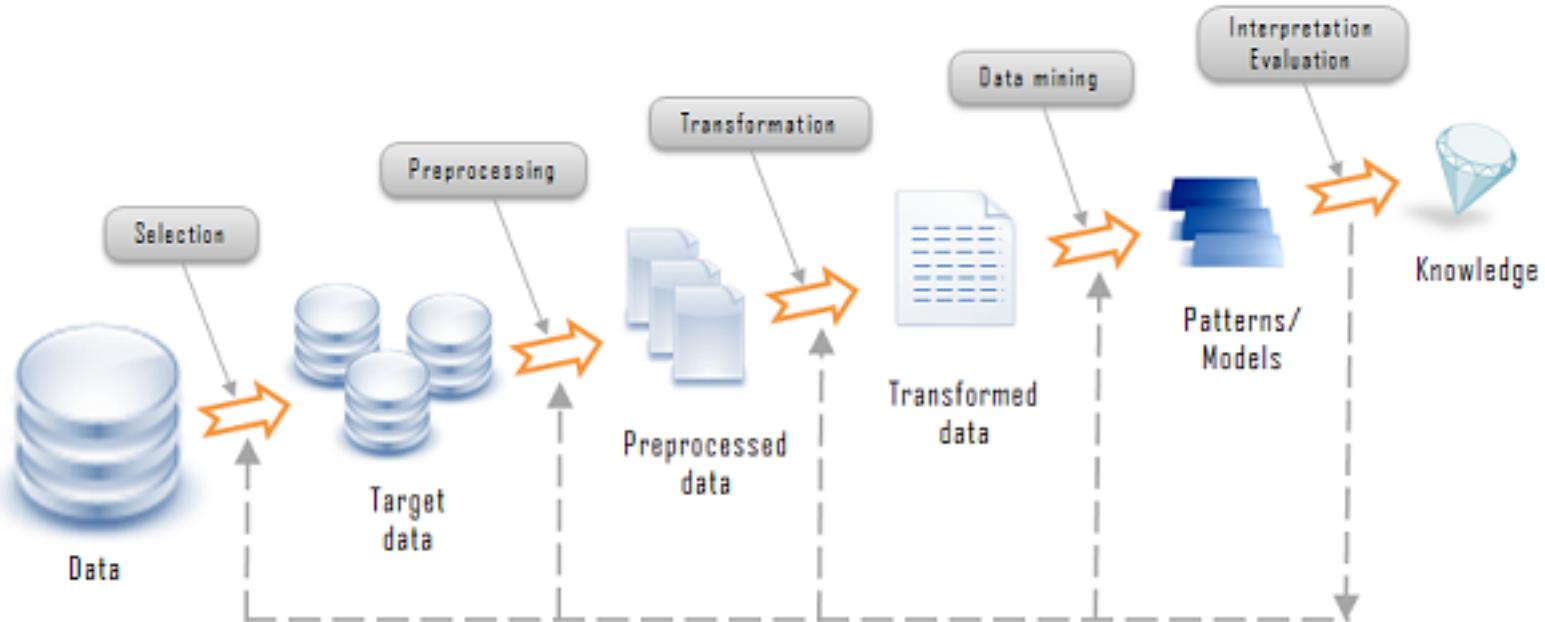
// Build the recommendation model using ALS
val rank = 10
val numIterations = 20
val model = ALS.train(ratings, rank, numIterations, 0.01)

// Evaluate the model on rating data
val usersProducts = ratings.map { case Rating(user, product, rate) =>
    (user, product)
}
val predictions =
    model.predict(usersProducts).map { case Rating(user, product, rate) =>
        ((user, product), rate)
}
val ratesAndPreds = ratings.map { case Rating(user, product, rate) =>
    ((user, product), rate)
}.join(predictions)
val MSE = ratesAndPreds.map { case ((user, product), (r1, r2)) =>
    val err = (r1 - r2)
    err * err
}.mean()
println("Mean Squared Error = " + MSE)
```

Synthèse

- Apprentissage supervisé
 - Classification
 - Arbre de décision
 - Classification Bayésienne naïve
 - Régression logistique
 - Support à vecteur de machine
 - Régression
 - Régression linéaire
 - Collaborative Filtering
 - Alternating Least Squares
 - Apprentissage supervisé
 - Clustering
 - K-plus proches voisins

Processus de découverte de la connaissance



Techniques de data mining

- Collaborative Filtering

- Objectif

- Prédire l'intérêt d'un utilisateur pour un item

- Prérequis

- Collecter des préférences utilisateurs du passé

- Clustering

- Regrouper les objets similaires

- Classification

- Placer les items dans des catégories prédéfinies

- Frequent Pattern Matching

- Rechercher les items qui se retrouvent souvent ensemble

- Dimensionality reduction

Recommandation : Collaborative Filtering

● Recommandation automatique

- Prévoir l'intérêt d'un utilisateur pour un item

User/Item	P1	P2	P3	P4
U1	5	4		1
U2	4			
U3			2	5
U4	1		2	
U5	1	5		2
U6		5	5	

- Objectif : Remplir les cases vides (mais pas toutes) pour “deviner” les préférences des utilisateurs
- Algorithmes : Alternating Least Squares

Clustering : Apprentissage non supervisé

- Recommandation automatique sans historique de comportement

Feature/Item	P1	P2	P3	P4
CPU	3.2	2.7	1.4	2.9
SSD	1	1	0	0
Disque	1To	512G	512G	1To
RAM	32Go	16Go	8Go	64G

- Objectif
 - Calculer la distance entre les items en pondérant les valeurs des features
 - Distance entre deux documents permet d'identifier des plagiats par exemple
- Algorithmes
 - K-means

Classification : Apprentissage supervisé

- Idem Clustering

- A l'exception que les clusters sont connus à l'avance

- Application

- Catégorisation de texte
 - Segmentation de marché

- Algorithmes

- Linear Support Vector Machines
 - Logistic Regression
 - Decision Trees
 - Naïve Bayes

Validation d'un algorithme

Feature/Item	CPU > 2.1	SSD	Disque >= 512	RAM >= 32Go	Class
P1	Oui	Oui	Oui	Oui	Station de travail
P3	Non	Non	Oui	Non	Bureautique
P4	Oui	Non	Non	Oui	Station de travail

Feature/Item	CPU > 2.1	SSD	Disque >= 512	RAM >= 32Go	Class
P2	Oui	Oui	Oui	Non	?

- Apprentissage avec le training set
- (Cross) Validation avec le test set
- Risques
 - Overfitting sur le training set

Préparation des données

- Choix des données

- Le choix des données de training et de tests s'appuie sur la connaissance métier

- Choix des critères

- Le choix des critères dépend également de la connaissance métier

- Choix des algorithmes

- Classification ? Régression ?
 - Label ou non ou mixte ?
 - Proportions de classes fortement asymétriques
 - L'objectif est de prédire une probabilité ? topN ?

- Tuning des paramètres des algorithmes en testant

- Tout automatiser pour avoir un historique précis des résultats obtenus