TSVD11

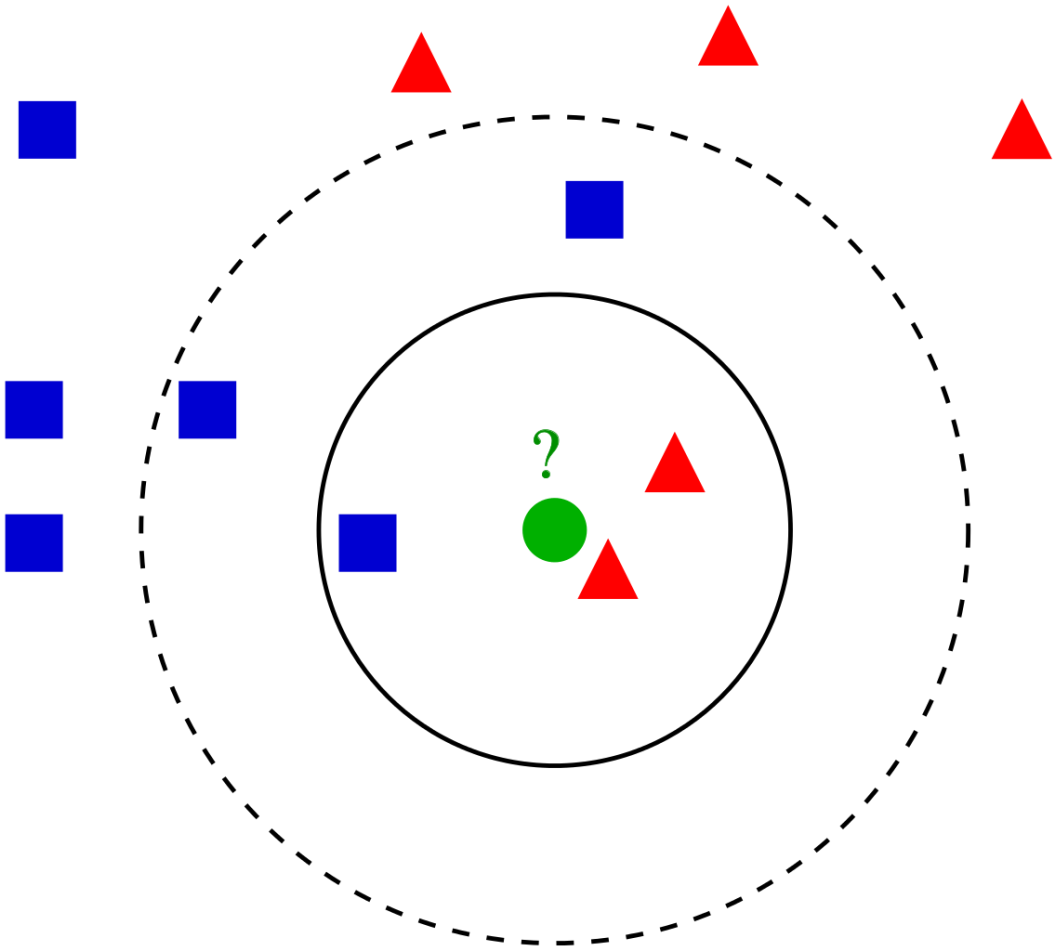
Technológie spracovania veľkých dát

Peter Bednár, Martin Sarnovský

Obsah

- kNN
- Naïve Bayes
- Lineárna regresia
- Odporúčacie mechanizmy - kolaboratívne filtrovanie
- Apache Mahout
- Spark MLLib

K-NN algoritmus



MapReduce k-NN

MapReduce k-NN

- Mapper
  - Vstup:
    - všetky vektory splitu datasetu
    - neoznačný príklad *p*
  - Výstup:
    - výpočet vzdialeností, nájdenie lokálnych *k* najbližších susedov
    - emituje *k* najbližších bodov k príkladu *p*
- Reducer
  - Vstup:
    - *Kľúč* – null; *hodnota* – lokálny susedia k príkladu *p*
  - Výstup:
    - agreguje k najbližích susedov z lokálnych *k* susedov
    - klasifikuje príklad *p*

Naïve Bayes

- Naive Bayes klasifikátor
- Každý príklad, vektor $x=(x_1, x_2, ..., x_n)$

- $P(Class_j|x) = \frac{P(x|Class_j) \times P(Class_j)}{P(x)}$

- $P(x|Class_j) = \prod_{k=1}^{n} P(xk|C_j)$ pravdepodobnosť z trénovacej množiny, ak A_k je kategorický, potom $P(xk|C_j)=\frac{s_{jk}}{S_i}$ kde s_{jk} je počet príkladov trénovacej množiny patriacich triede C_j, pre ktoré $A_k = x_k$, S_i je počet vzoriek z trénovacej množiny patriacich triede C_j
- $P(Class_j)$ - apriórna pravdepodobnosť, s akou ľubovoľný prvok patrí do triedy j

Naïve Bayes - MapReduce

- Map
  - Vstupné dáta z podmnožín dát
  - Map transformuje dáta do formy kľúč - hodnota
  - 3 typy kľúčov:
    - Kľúč pre $P(x|Class_j)$, hodnotou je potom hodnota pravdepodobnosti
    - Kľúč pre $P(Class_j)$, hodnotou je hodnota pravdepodobnosti
    - $P(x)$ - konštanta
- Reduce
  - Agreguje hodnoty pre jednotlivé kľúče
  - Spočíta celkové podmienené pravdepodobnosti

MapReduce algoritmy - zhrnutie

- Ne-iteratívne algoritmy – ”perfect fit”
  - Canopy clustering, Naïve Bayes, k-NN
- Algoritmy realtívne málo iterujúce OK
  - KMeans, LDA
- Pre niektoré – MapReduce nevhodné (iné alternatívy – MPI, in-memory, atď.)
  - Support Vector Machines

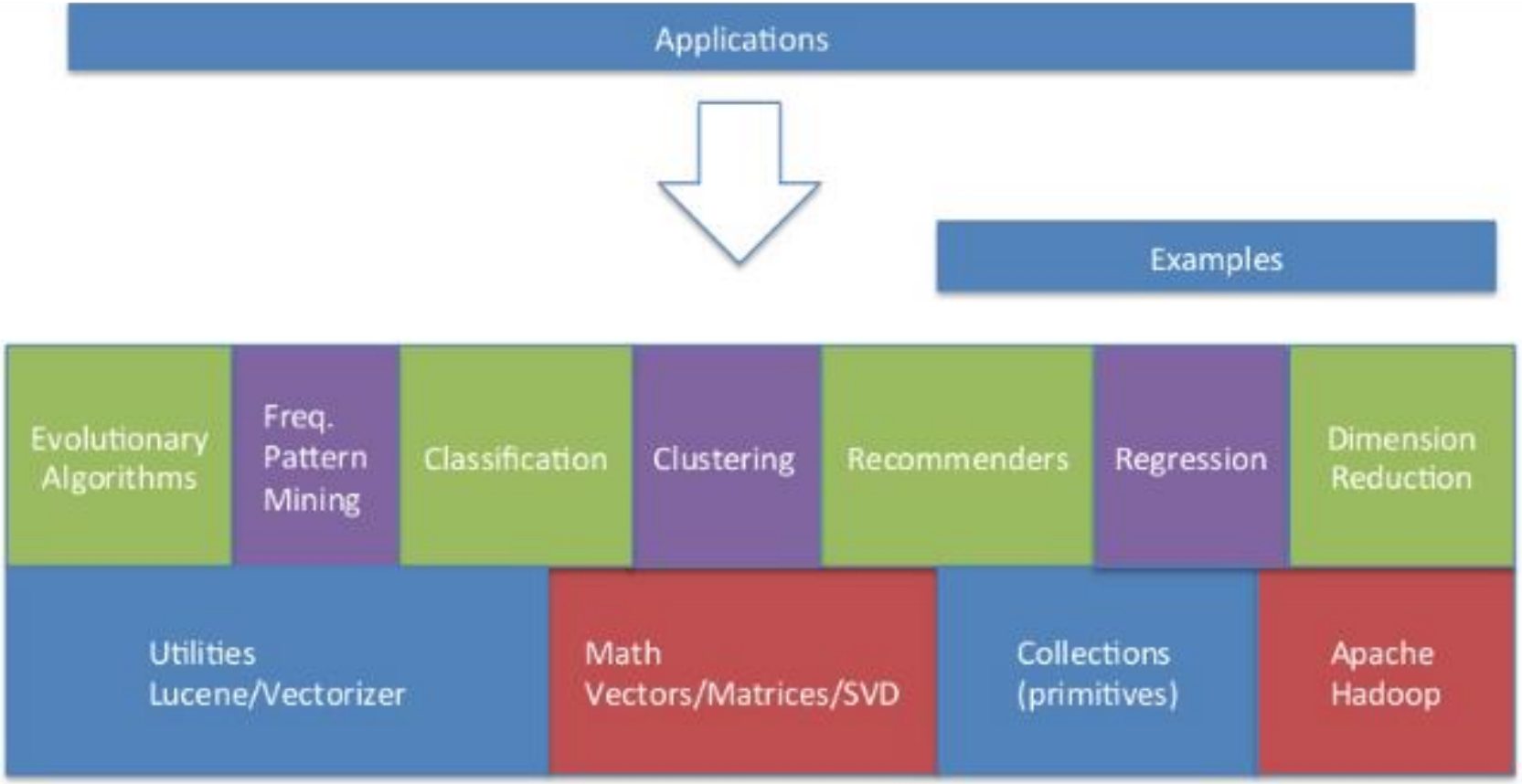
Mahout

- Knižnica pre strojové učenie na platforme Hadoop
- Pôvodne Lucene pod-projekt
- Distribuované MapReduce implementácie algoritmov strojového učenia
- Napísaná v Jave
- Využíva benefity distribuovanej platformy Hadoop pre ML úlohy
- Implementuje algoritmy a podporné nástroje pre úlohy
  - Klasifikácie
  - Zhlukovania
  - Generovanie odporúčaní
- Poskytuje aj Java knižnice pre matematické operácie (štatistika, lineárna algebra)

Mahout

- Modely vyžadujú ako vstup *SequenceFile*
- Nutnosť konvertovať dataset do tohto formátu (konverzia = MapReduce úloha)
- Potreba konvertovať dáta – duplikácia dát na klastri
- Konverzia dát – tým, že sa jedná o MapReduce job, nutnosť čítať a opäť zapísať dáta do HDFS
- Iteratívne modely (napr. K-Means) – MapReduce implementácie – vyžadujú častý zápis a čítanie z disku -> relatívne pomalšie oproti in-memory prístupom

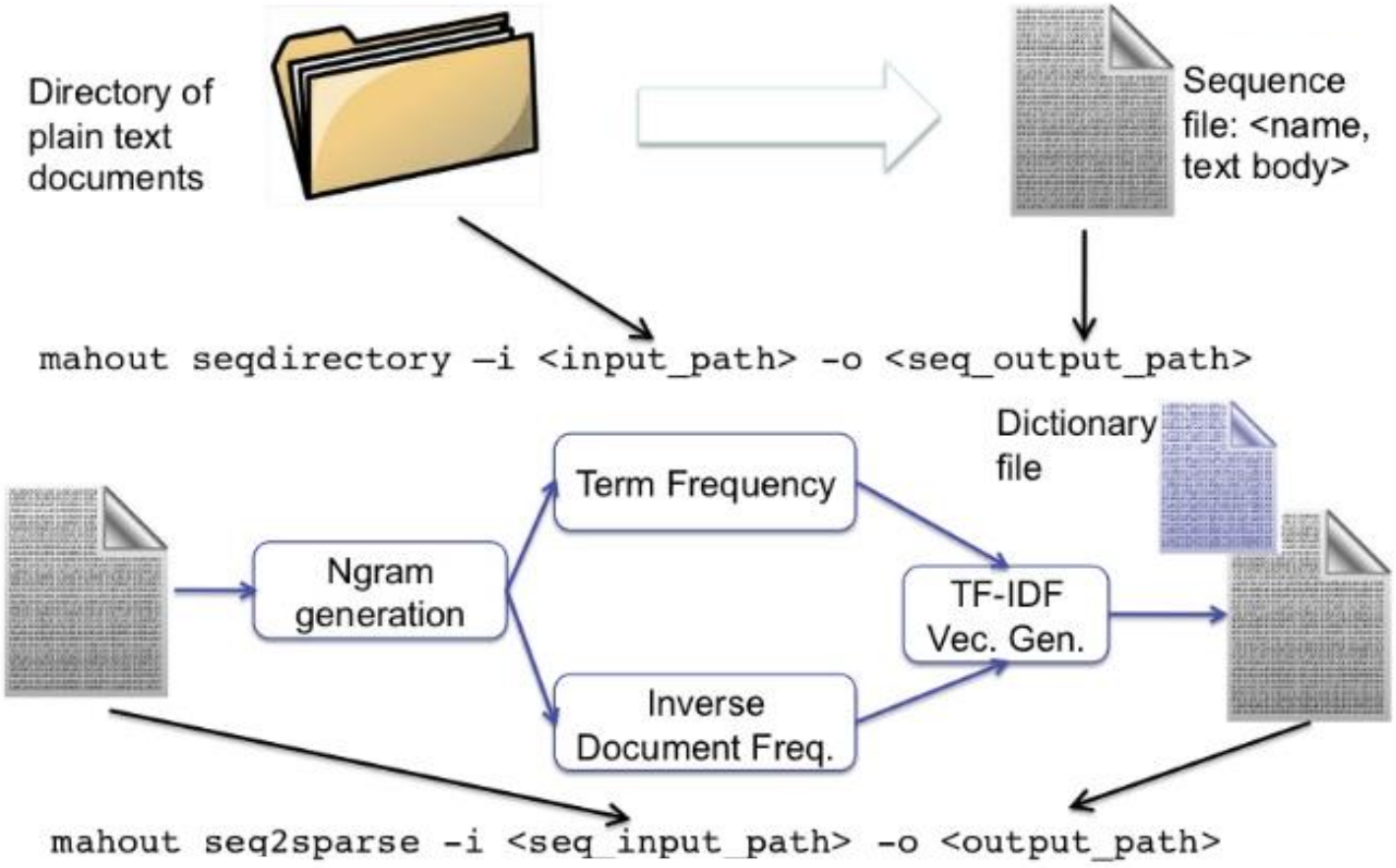
Mahout Architektúra



Vektorizácia

- Transformácia vstupných dát do vektorizovanej podoby – vykonávané ako MapReduce job
- Vstupné formáty pre jednotlivé modely stále nie uniformné naprieč Mahoutom
- Väčšina algoritmov očakáva trénovacie dáta transformované na *SequenceFile*
  - *SequenceFile(WritableComparable, VectorWritable)*
- Pre spracovanie textových dokumentov určitá podpora
  - Môžeme použiť Mahout API pre generovanie TFIDF reprezentácie textov

Vektorizácia textových dokumentov



Mahout vytvorenie modelu

- Spustenie k-means ako MapReduce jobu z command line

```
mahout kmeans
   -i <adresár vstupných vektorov>
   -c <adresár zhlukov >
   -o <výstupný adresár>
   -k <parameter modelu -k>
   -dm <metrika vzdialenosti>
   -x <maximálny počet iterácií>
   -ow <možnsť prepísať výstup, ak existuje>
   -xm <spôsob spustenia: sequential alebo mapreduce>
```

- Použitie v aplikácii (Mahout API)

```
KMeansDriver.runJob("testdata", "output/clusters-0", "output",
EuclideanDistanceMeasure.class.getName(), "0.001", "10", true);
```

Modely - zhlukovanie

- LDA (topic models)
- K-Means (seq, mapreduce)
- Fuzzy K-Means (seq, mapreduce)
- Streaming K-Means (seq, mapreduce)
- Canopy clustering (seq, mapreduce, streaming)
- Spectral clustering (mapreduce)

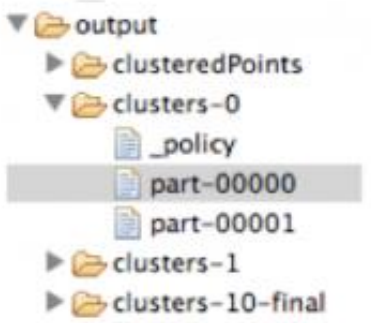
Modely - klasifikátory

- Logistická regresia (seq)
- Support Vector Machines
- Random Forests (mapreduce)
- Naive Bayes (mapreduce)
- Multi-level Perceptron (seq)
- Hidden Markov Models (seq)

Ostatné modely v Mahout

- Odporúčacie systémy v Mahout
  - User-Based collaborative filtering (seq)
  - Item-Based collaborative filtering (seq, mapreduce)
  - ALS (seq, mapreduce)
- Modelovanie témy
  - LDA (seq, mapreduce)
- Redukcia dimenzionality
  - SVD (seq)
  - PCA (seq, mapreduce)

Používanie a vyhodnotenie modelu



- Po vytvorení modelu
  - Štruktúra výstupného adresára
  - Špecifická pre jednotlivé modely
- Nástroje pre prístup k modelu a vyhodnotenie
- Napr. V prípade zhlukovania KMeans *ClusterDumper* a *DisplayKmeans* tooly umožňujúce vizualizovať obsah zhlukov

CL-0 { n=116 c=29.922, 30.407, 30.373, 30.094, 29.886, 29.937, 29.751, 30.054, 30.039, r=[3.463, 3.351, 3.452, 3.438, 3.371, 3.569, 3.253, 3.531, 3.439, 3.472,]}

Spark MLlib

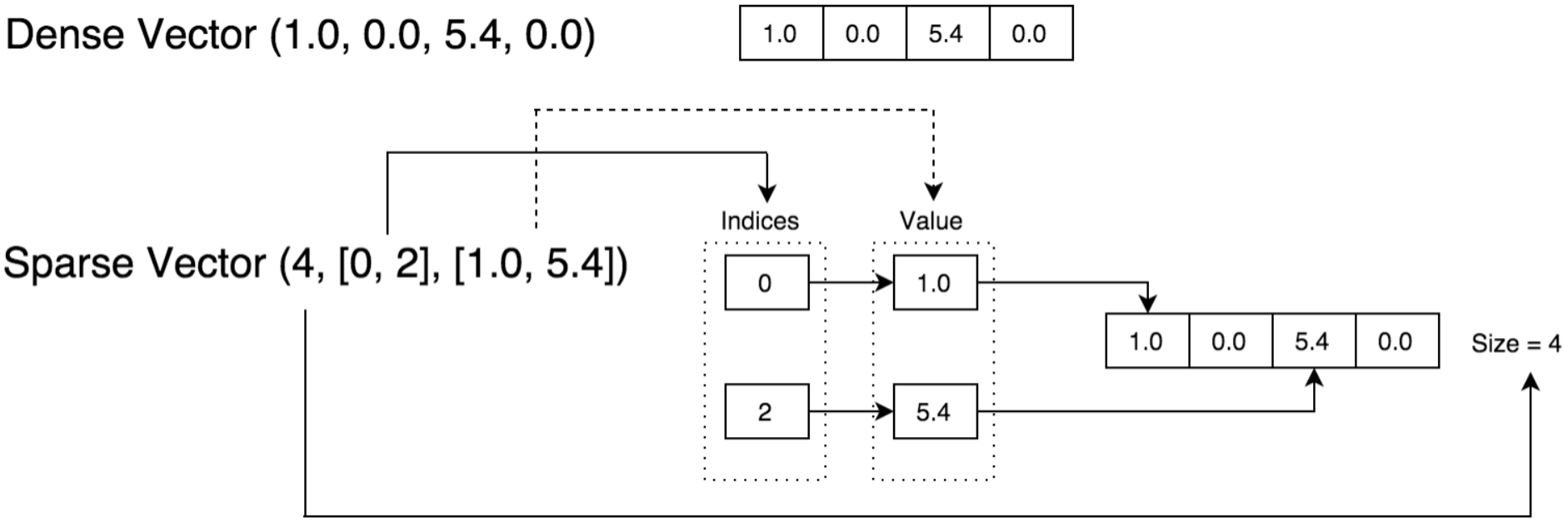
- MLlib – Machine Learning library
- Spark podprojekt obsahujúci implementácie metód strojového učenia na platforme Spark
- Prvýraz súčasťou Spark vo verzii 0.8 (Sept. 2013)
- Štandardný komponent Spark stack-u
- Dobrá integrácia s ostatnými komponentami Spark stack-u (Streaming, Spark SQL, GraphX)
- Natívne implementovaný v Scale
- API pre Java aj Python
- Vďaka Spark platforme – porovnateľný, resp. Výkonnejší ako Mahout/MapReduce na Hadoop platforme

Modely v Mllib

- Klasifikácia/predikcia:
  - Logistická regresia, support vector machine (SVM), Naïve Bayes, stromy, Random Forests, Boosting
- Kolaboratívne filtrovanie: alternating least squares (ALS)
- Zhlukovanie: k-means (niekoľko modifikácií, vrátane streaming)
- Dekompozícia: singular value decomposition (SVD), principal component analysis (PCA)
- Utility: metriky pre evaluáciu, testovanie, štatistické výpočty atď.

MLlib - Dátové typy

- Dense Vector, Sparse Vector, Labeled Point



- posP = LabeledPoint(1.0, [1.0, 0.0, 5.4, 0.0])
- negP = LabeledPoint(0.0, SparseVector(4.0, [0,2], [1.0, 5.4]))

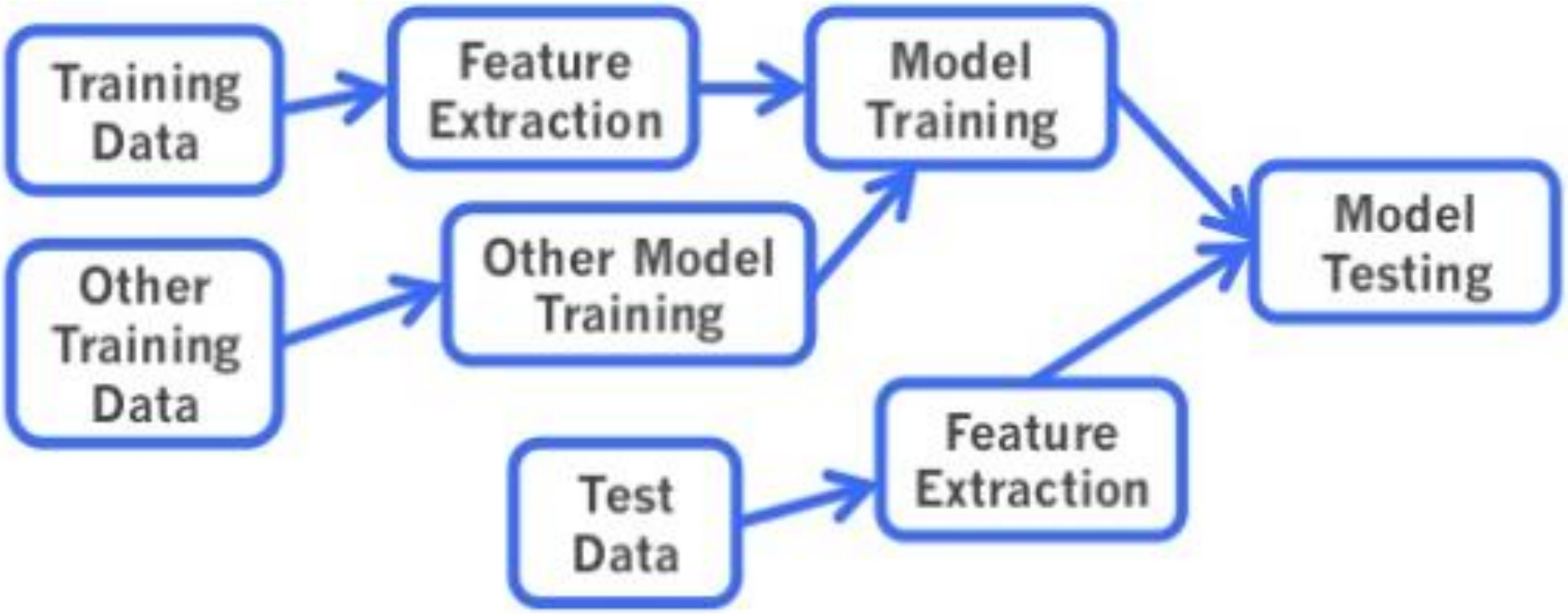
ML Pipelines

- Nové, high-level API pre MLlib (od verzie 1.2 v alpha verzii)
- Zjednodušená tvorba jednotlivých úloh v úlohách strojového učenia ako workflows
- Definovaním postupnosti krokov – tzn. stages
- Hlavné koncepty ML Pipelines:
  - Data Frames
  - Transformer
  - Estimator
  - Pipeline
  - Parameter

ML Pipelines

- DataFrame: používajú sa dátové rámce zo Spark SQL
- Transformer: je to algoritmus, ktorý transformuje jeden dátový rámec na iný,
  - napr. čítanie z textového súboru a zapisovanie do stĺpcov dátového rámca
  - napr. model strojového učenia je Transformer ktorý transformuje dátový rámec s vstupnými dátami na dátový rámec predikcií
- Estimator: je algoritmus, ktorý môžeme použiť na dátový rámec a vyprodukuje transformer, napr. učiaci sa algoritmus je Estimator, ktorý trénujeme na dátovom rámci a výsledkom je model
- Pipeline: je to zreťazenie viacerých Transformers a Estimators, ktoré tak formujú pracovný tok ML úlohy
- Parameter: všetky Transformers a Estimators zdieľajú spoločné API pre špecifikáciu parametrov

ML Pipelines



ML pipelines - príklad

- Deklaruje “stages” (tokenizer, TF extraktor, Regresný model), nastaví parametre a zreťazí do jedného objektu

```
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import HashingTF, Tokenizer

# trenovacia mnozina

training = spark.createDataFrame([(0,"a", 1.0),(1,"b", 0.0), (2,"spark", 1.0)],["id", "text", "label"])

# Nakonfigurujeme ML pipeline, ktora pozostava z 3 stages: tokenizer, hashingTF a lr

tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.001)

pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

# Pouzujeme tuto pipeline na trenovacie dokumenty

model = pipeline.fit(training)
```