Machine Learning voorbeeld met Apache Spark's MLlib

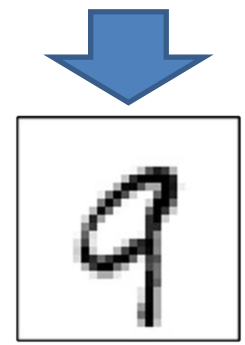
System Requirements

- Java
- Maven
- Scala
- Apache Spark:
  - Meerdere versies beschikbaar op website van Apache Spark. Kale source code versie moet eerst zelf gebouwd worden met een buildtool e.g. Maven. De andere versies zijn allemaal pre-built voor een bepaalde Hadoop versie. Welke versie van de pre-builts je gebruikt maakt niet uit als je Spark in pseudo cluster mode gaat draaien)

Probleem, We hebben

- Een dataset van 42000 ‘tekeningen’ van cijfers
- Eerste kolom bij een rij vertelt je welk cijfer getekend is.
- De rest van alle kolommen zijn greyvalues van alle pixels in een plaatje (0 is wit, 255 is zwart).

9,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,7,59,169,228,197,75,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,59,209,254,254,254,254,222,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,81,243,2
47,171,87,89,254,232,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,17,139,254,177,48,0,
0,158,254,222,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,130,254,165,7,0,0,43,253,254,
151,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,68,247,145,5,0,0,0,152,255,202,50,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,40,217,92,0,0,0,0,0,211,252,49,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,118,198,3,0,0,0,0,0,213,215,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,154,32,0,0,
0,0,0,10,213,114,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,34,187,23,0,0,0,33,164,219,2
51,127,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,49,254,49,0,0,75,191,254,204,246,119,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,37,246,215,146,203,248,254,157,21,193,174,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,79,199,254,254,145,75,3,0,182,141,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,25,85,85,22,0,0,0,239,138,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,239,159,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,29,
246,85,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,33,247,85,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,100,252,85,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,146,254,85,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,74,250,59
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

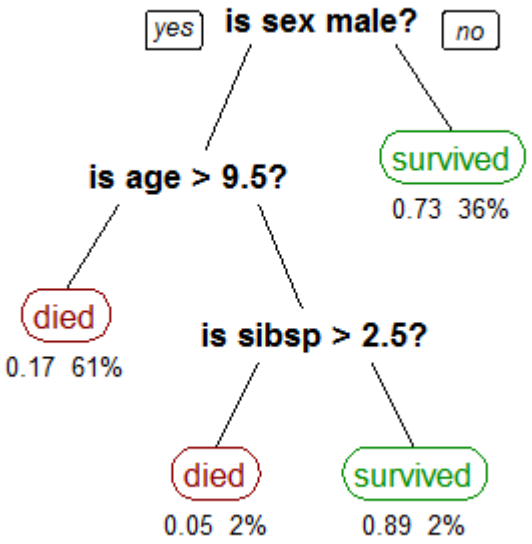


Probleem, We willen

- Gegeven een plaatje van een cijfer, willen we geautomatiseerd kunnen voorspellen welk cijfer het is.
- Aan de hand van data een model ‘trainen’ en dat model gebruiken om te voorspellen.
- Dit is een Classificatie probleem, waarvoor Decision Trees uitzonderlijk geschikt zijn, en hun uitbreiding Random Forest misschien nog wel meer.

Decision Tree

- Classificatie model.
- Afhankelijk van input loop je door de tree om bij de meest waarschijnlijke optie uit te komen.
- Model wordt getraind aan de hand van ‘information gain’: het attribuut dat de data het duidelijkste (afhankelijk van een gespecificeerde informatie gain maat) opdeelt in de verschillende klasses komt bovenaan in de beslisboom. Dit process wordt herhaald tot een maximale diepte wordt bereikt, of alle attributen gebruikt zijn.



Random Forest

- Een ‘ensemble’ van *n* decision trees, elk getraind op de data, met slechts een random subset van de kolommen (e.g. de eerste 100 kolommen ipv alle 784).
- Alle *n* modellen een voorspelling laten doen.
- De meest voorkomende voorspelling is de gekozen voorspelling van het ensemble.

Code draaien in Spark

- Code beschikbaar op https://github.com/CasperKoning/digit-classification
- Pas projectRoot variabele in code aan (ik was te lui om het netjes op te lossen)
- Eerst een *mvn clean package* van DigitClassification project.
- Dan, op CLI:

  <SPARK_HOME>\bin\spark-submit [options] <app jar>
- Bijvoorbeeld:

C:\spark-1.2.1\bin\spark-submit --master local[8] --class nl.ordina.decisiontree.DecisionTreeApp C:\digit-classification-1.0-SNAPSHOT.jar

- Om spark in pseudo cluster modus met acht cores de applicatie DecisionTreeApp.scala in digit-classification-1.0-SNAPSHOT.jar uit te laten voeren.

Resultaat

- DecisionTreeApp:
  - “Correct voorspeld: 85,52%”
- RandomForestApp:
  - “Correct voorspeld: 96,34% “