Klasifikasi Sederhana dengan Machine Learning

Alfan F. Wicaksono
Fakultas Ilmu Komputer
Universitas Indonesia

Tool yang Kita Gunakan

- Scikit-Learn (http://scikit-learn.org/stable/)

- Course ini hanya ditujukan untuk peserta yang "baru ingin belajar Machine Learning".

Data Mining

- Data Mining : the science of extracting useful knowledge from a huge data repositories. (ACM SIGKDD)

ACM SIGKDD : http://www.kdd.org/curriculum/index.html

Common Tasks

- **Classification**
- Clustering
- Regression
- Summarization
- Association Rules Mining
- Etc.

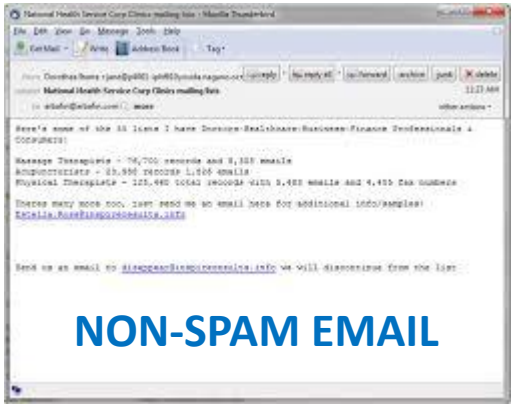
Klasifikasi

- Memberi label sebuah objek secara otomatis berdasarkan contoh-contoh data yang sudah diberikan sebelumnya.

- Contoh:
  - Klasifikasi apakah email yang masuk SPAM atau BUKAN SPAM ?
  - Tentukan apakah seseorang akan menderita penyakit jantung jika diketahui umur, detak jantung, dan tekanan darahnya.

Klasifikasi: Contoh 1

SPAM EMAIL



Kita ingin mengetahui apakah email berikut SPAM ?

Model Klasifikasi

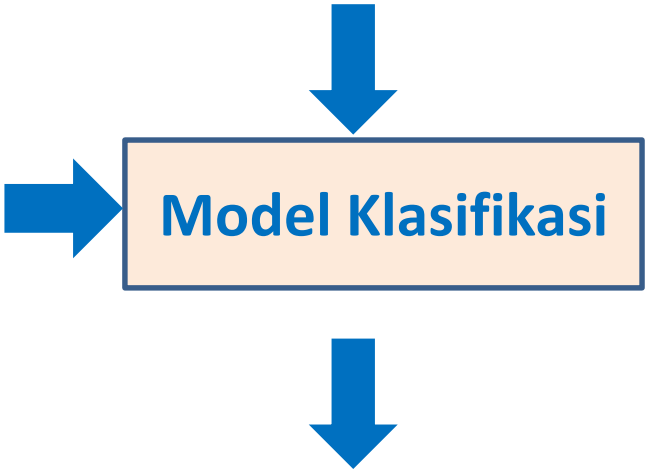
SPAM ? atau BUKAN SPAM ?

Klasifikasi: Contoh 2

Training data (sample)

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Rainy	Cool	Normal	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Mild	High	False	No
Rainy	Mild	High	True	No
Sunny	Mild	Normal	True	Yes

Seandainya, hari ini **hujan**, **panas**, **sangat lembab**, dan **tidak berangin**



Play Outside or Not ?

Model Klasifikasi

- Rule-based Approach
- Data-driven Approach (Machine Learning)
  - Infer rule secara otomatis dari data masa lampau yang berukuran besar

Machine Learning

- Machine learning is the science of getting computers to act without being explicitly programmed.

https://www.coursera.org/learn/machine-learning, by Stanford University

Machine Learning Model untuk Klasifikasi

- Naive Bayes
- Logistic Regression (Maximum Entropy)
- Support Vector Machine
- Nearest Neighbors
- Decision Trees
- Ensemble Methods
  - Random Forest, AdaBoost, Gradient Tree Boosting
- Neural Networks (MLP, Deep Learning)

State-of-the-art

(model yang sedang terkenal saat ini, Agustus 2016)

- XGBoost (Extreme Gradient Boosting)
  - https://github.com/dmlc/xgboost
- Convolutional Neural Networks (CNNs)
  - Biasanya jika dataset berupa Gambar/Image
- Recurrent Neural Networks (RNNs)
  - Dan variannya seperti LSTM (Long-Short Term Memory) dan GRU (Gated Recurrent Unit)
  - Biasanya jika dataset berupa Teks (atau jika **data memiliki aspek Sequence**)
- Dan arsitektur Deep Learning yang lain

- Anda memang bisa saja menggunakan algoritma-algoritma ML tersebut sebagai **Black Box** untuk Classification Task Anda.
  - Menggunakan tools yang tersedia di Internet

- Tetapi, akan lebih baik jika Anda memahami beberapa dari algoritma machine learning tersebut (setidak pada high-level view).

Sekilas: Naive Bayes

- Misal, **X** adalah input (direpresentasikan sebagai **fitur-fitur**), dan **y** adalah label/kelas.

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Sekilas: Logistic Regression

- Misal, **X** adalah input (**fitur-fitur**), **w** adalah bobot yang berasosiasi di setiap fitur, **y** adalah label.
- Cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

- Training: diberikan data, cari **w** sehingga cost function minimal !

Fitur

- Fitur: hal-hal yang menjadi karakter sebuah objek
- Fitur harus diskriminatif
- Pemilihan fitur adalah yang penting !
- Sehebat apapun machine learning algorithm yang kita gunakan, jika fiturnya tidak diskriminatif, performa model klasifikasi kita tidak akan bagus.

Contoh mencari Fitur

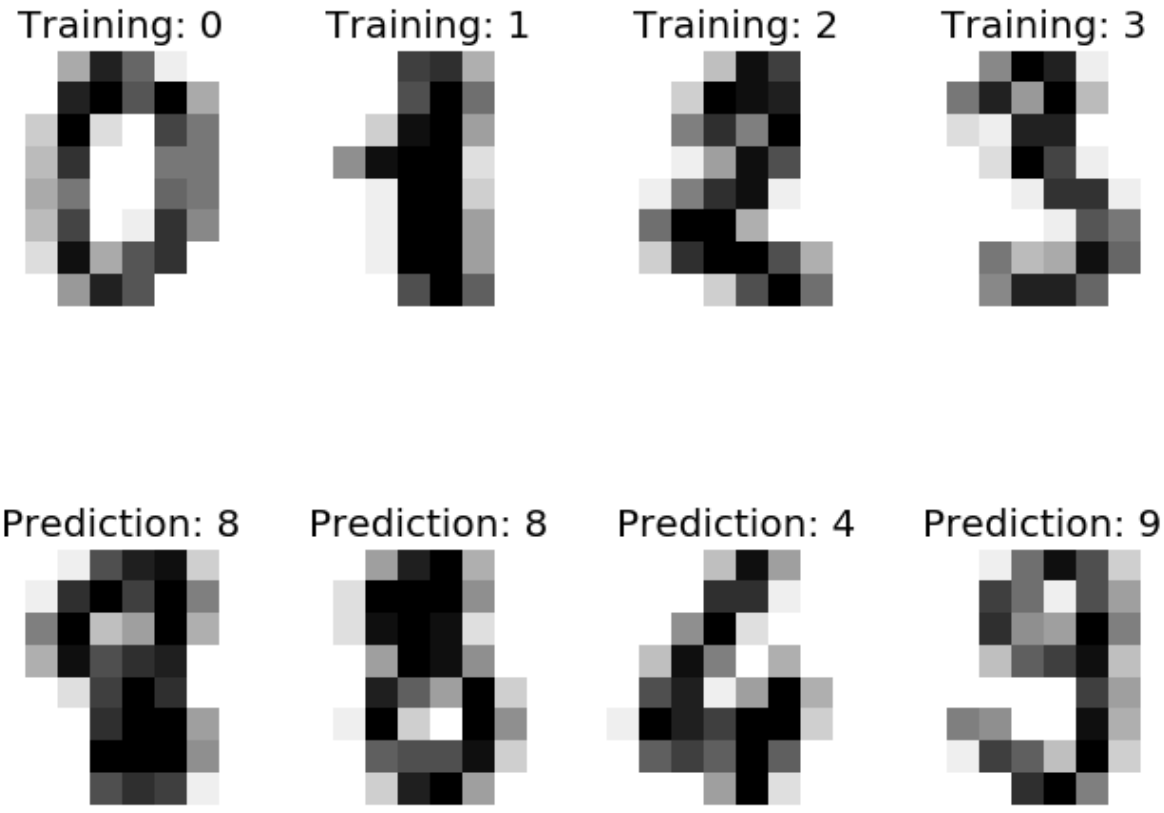
- Task: klasifikasi jenis kelamin/gender
- Objek: Manusia

- Fitur1: umur, warna baju, ukuran sepatu
- Fitur2: warna suara, panjang rambut, apakah memakai Rok?

Manakah kelompok fitur yang diskriminatif?

Hands On: Hand-written Digit Classification

Task: Hand-written Digit Classification



Yang akan kita lakukan

- Load dataset
- Preprocessing dataset (scaling)
- Split dataset: Training set, Testing set
- Pilih model machine learning
- Latih model tersebut dengan training set
- Evaluasi performa model terhadap testing set

Step 0: Import Package Scikit-Learn yang dibutuhkan

```
from sklearn.datasets import load_svmlight_file

from sklearn import preprocessing

from sklearn.cross_validation import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report,
confusion_matrix

import numpy as np
```

Dataset

- Dataset MNIST yang sangat terkenal
- http://yann.lecun.com/exdb/mnist/

Format Dataset **Libsvm**

```
5 153:3 154:18 155:18 156:18 ...
0 128:51 129:159 130:253 ...
4 161:67 162:232 163:39 173:62 ...
1 159:124 160:253 161:255 162:63 ...
9 209:55 210:148 211:210 ...
2 156:13 157:25 158:100 ...
...
```

Step 1: Dataset

- Silakan lihat file Dataset1.txt
- Di tutorial kali ini, kita asumsikan fitur dari objek yang ingin kita klasifikasikan sudah di-ekstrak.
- Fitur berupa spektrum piksel.
  - Gambar berukuran 28x28 piksel. Jadi adalah 784 fitur
  - Nilai untuk setiap fitur berkisal 0 – 255. 0 artinya hitam dan 255 artinya putih.

Step 1: Dataset

- Kode untuk load dataset kita ke memory

```
#load dataset in svmlib format
X, y = load_svmlight_file("dataset1.txt")

#X is scipy.sparse CSR matrix,
#we need to convert it to numpy array
X = X.toarray()
```

Step 2: Preprocessing Data

Proses mengubah raw feature vectors biasa menjadi bentuk yang cocok untuk beberapa model machine learning.

- Standardization
  - Zero mean & unit variance
- Normalization
- Binarization
- Imputation of missing values
- dsb

Lihat http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing

Standardization I
(**zero mean and unit variance**)

```
>>> from sklearn import preprocessing
>>> import numpy as np
>>> X = np.array([[ 1., -1.,  2.],
...               [ 2.,  0.,  0.],
...               [ 0.,  1., -1.]])
>>> X_scaled = preprocessing.scale(X)

>>> X_scaled
array([[ 0.  ..., -1.22...,  1.33...],
       [ 1.22...,  0.  ..., -0.26...],
       [-1.22...,  1.22..., -1.06...]])
```

Standardization II
(**scaling to [0,1]**)

```
>>> X_train = np.array([[ 1., -1.,  2.],
...                     [ 2.,  0.,  0.],
...                     [ 0.,  1., -1.]])
...
>>> min_max_scaler = preprocessing.MinMaxScaler()
>>> X_train_minmax =
min_max_scaler.fit_transform(X_train)
>>> X_train_minmax
array([[ 0.5       ,  0.        ,  1.        ],
       [ 1.        ,  0.5       ,  0.33333333],
       [ 0.        ,  1.        ,  0.        ]])
```

Step 2: Preprocessing Data

Kita akan scaling dataset kita, sehingga setiap fitur mempunyai nilai [0,1]

```
#scaling to [0,1]
min_max_scaler = preprocessing.MinMaxScaler()
X_scaled = min_max_scaler.fit_transform(X)
```

Step 3: Split Train-Test

- Training Data digunakan untuk melatih model
- Testing Data digunakan untuk menguji model yang sudah dilatih sebelumnya

Step 3: Split Train-Test

- Dari dataset1.txt kita, ada 5000 instances
- Kita akan bagi 70%/**30%** untuk training data dan testing data.

```
#split train-testing
X_train, X_test, y_train, y_test =
     train_test_split(X_scaled, y, test_size=0.3)
```

Step 4: Pilih Algoritma ML

- Kita pilih Logistic Regression

```
#model: logreg
classifier = LogisticRegression()
```

Step 4: Latih Model (Training)

- Fase Training / Parameter estimation
- Di fase inilah, model kita belajar dari data yang diberikan

```
#split train-testing
classifier.fit(X_train, y_train)
```

Step 5: Prediksi Testing Data

- Kita coba beri label test data menggunakan model kita
- Testing data sudah punya label yang benar untuk judgment nantinya

```
#predict testing data
y_predict = classifier.predict(X_test)
```

Step 6: Evaluasi Model

- Kita perlu menilai seberapa bagus model klasifikasi kita
- Penilaian didasarkan pada gold-standard yang ada pada testing data

- Metrics yang sering digunakan:
  - Accuracy
  - Precision
  - Recall
  - F1

Step 6: Evaluasi Model

		(gold standard) K1	(gold standard) K2
Computer (Model)	K1	A	B
	K2	C	D

- Accuracy = (A + D) / (A + B + C + D)
- Precision K1 = A / (A + B)
- Recall K1 = A / (A + C)
- Precision K2 = D / (C + D)
- Recall K2 = D / (B + D)

Step 6: Evaluasi Model

- Precision, Recall, F1-score untuk setiap label (0 - 9)

```
#reporting classification results on
#testing data (performance)
print(classification_report(y_test, y_predict))
```

Step 6: Evaluasi Model

- Confusion Matrix

```
#reporting confusion matrix
print(confusion_matrix(y_test, y_predict))
```

Ingin Mencoba Model yang Lain ?

Naive Bayes

Tambahkan kode pada **step 0** (import model Naive Bayes):

```
from sklearn.naive_bayes import GaussianNB
```

Ganti Kode pada **step 4**:

```
classifier = GaussianNB()
```

Kita menggunakan Gaussian Naive Bayes karena nilai fitur kita adalah **real-valued [0,1]**
Detail: http://scikit-learn.org/stable/modules/naive_bayes.html

SVM (asumsi Linearly Separable)

Tambahkan kode pada **step 0** (import model SVM):

```
from sklearn.svm import SVC
```

Ganti Kode pada **step 4**:

```
classifier = SVC(kernel='linear')
```

Detail: http://scikit-learn.org/stable/modules/svm.html

SVM (Non-Linearly Separable)

Tambahkan kode pada **step 0** (import model SVM):

```
from sklearn.svm import SVC
```

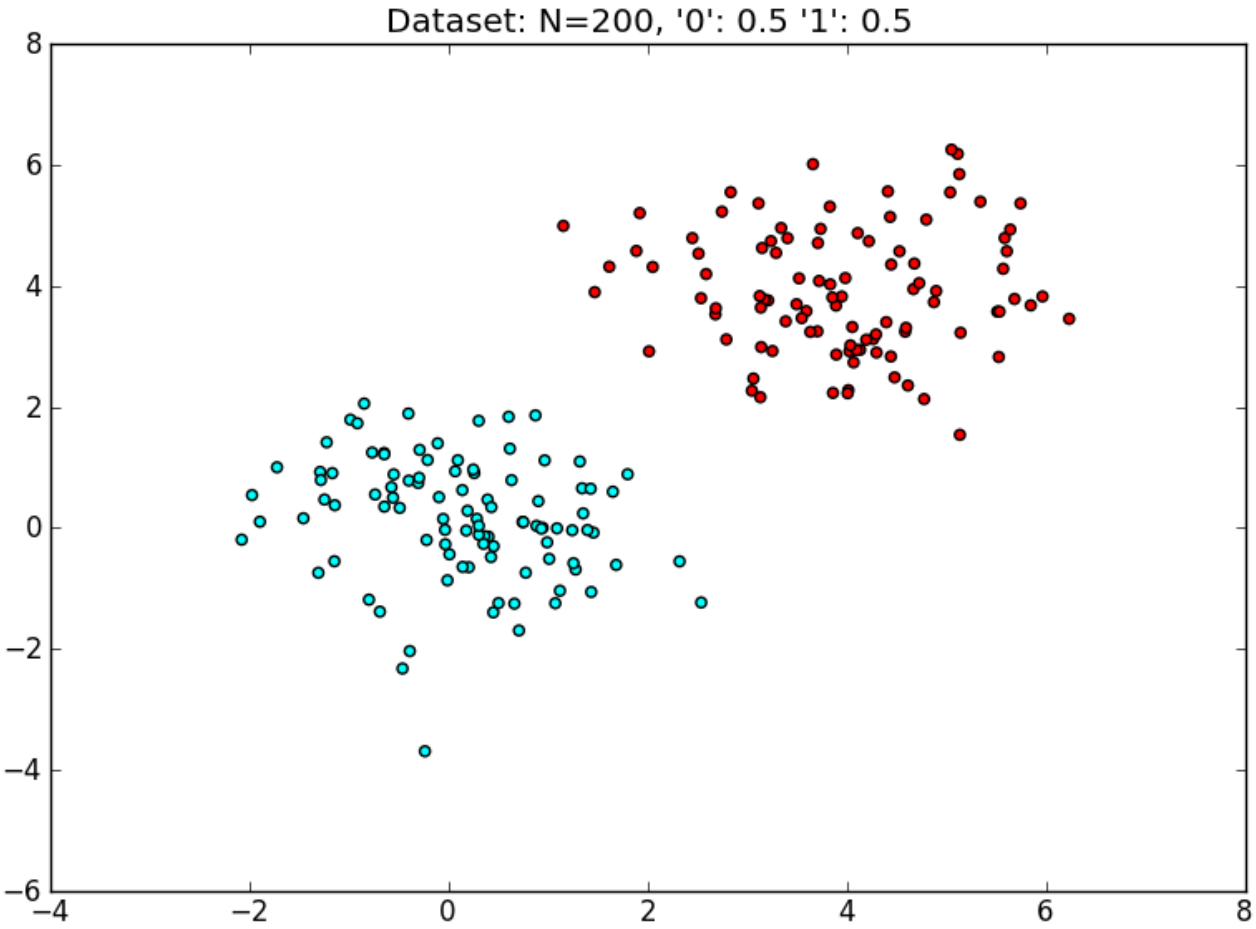
Ganti Kode pada **step 4**:

```
classifier = SVC(kernel='rbf')
```

Menggunakan Kernal Function RBF

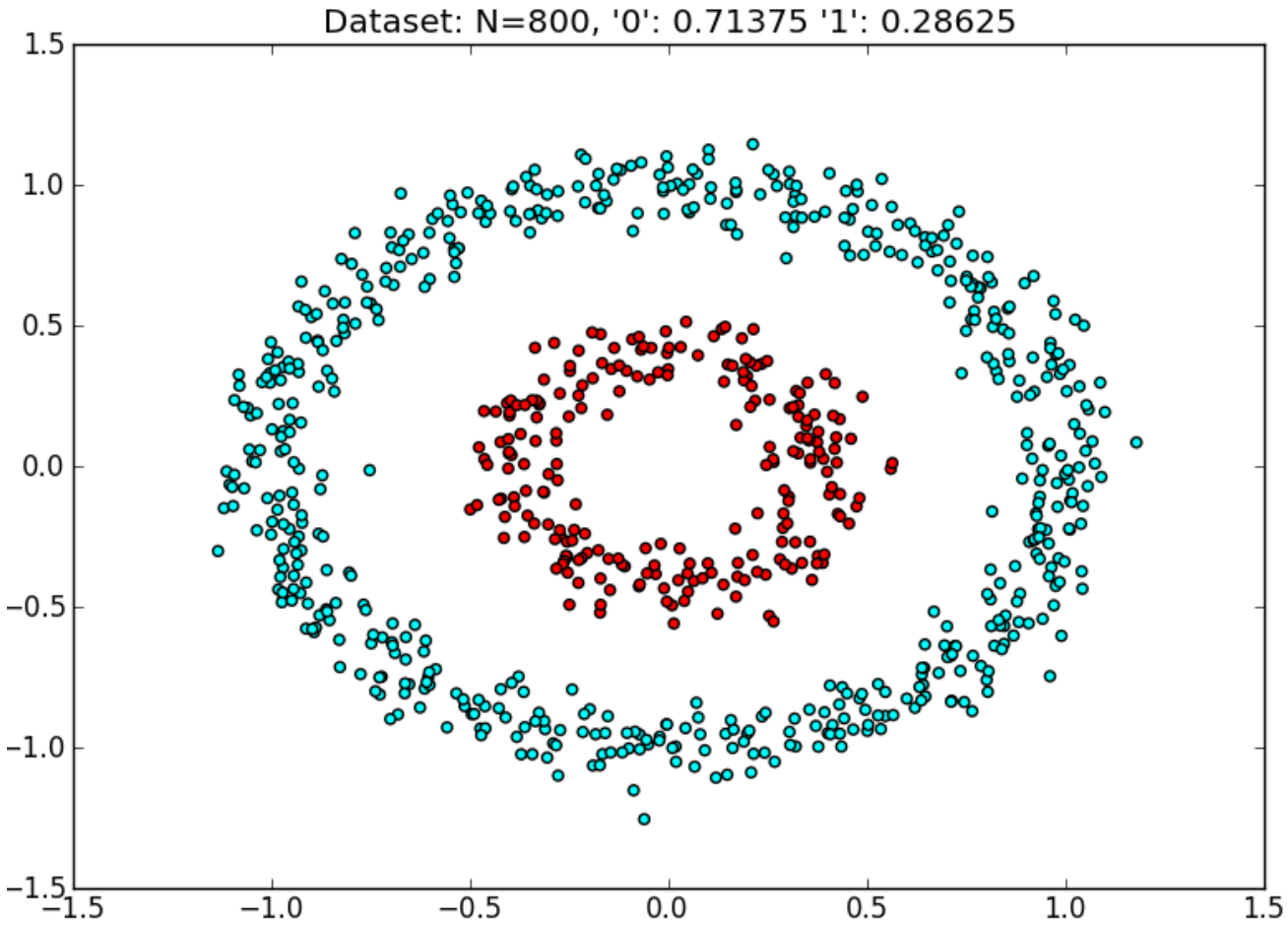
Detail: http://scikit-learn.org/stable/modules/svm.html

Linearly Separable Data



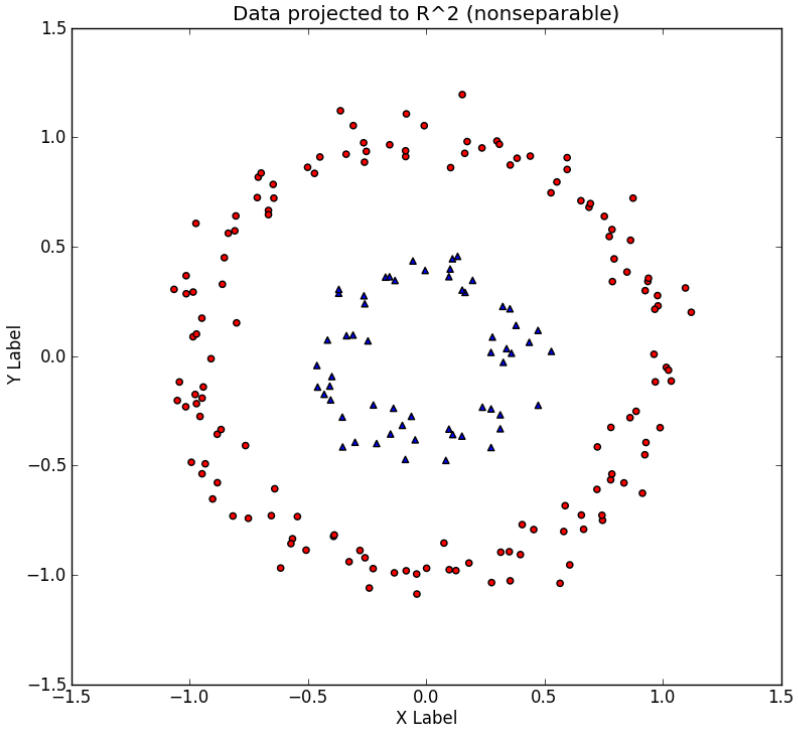
Detail: http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

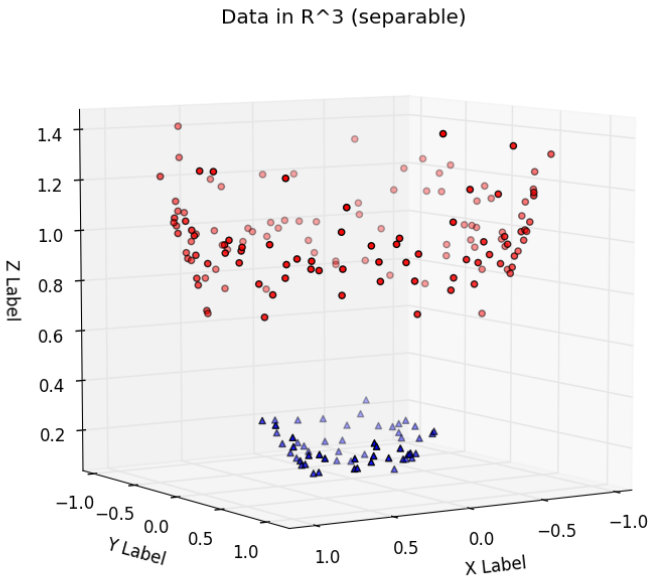
Non-Linearly Separable Data



Detail: http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Non-Linearly Separable Data?





$$[x_1, x_2] = [x_1, x_2, {x_1}^2 + {x_2}^2]$$

Bisa jadi mereka separable di dimensi lain yang lebih tinggi !

Detail: http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Ensemble: Gradient Tree Boosting

Tambahkan kode pada **step 0** (import model GTB):

```
from sklearn.ensemble import GradientBoostingClassifier
```

Ganti Kode pada **step 4**:

```
classifier = GradientBoostingClassifier(n_estimators=100)
```

Detail: http://scikit-learn.org/stable/modules/ensemble.html

Model yang lain

- Silakan lihat:

http://scikit-learn.org/stable/supervised_learning.html#supervised-learning

Feature Selection

Feature Selection, Why?

- Tidak semua fitur yang kita gunakan mempunyai nilai diskriminatif yang baik !
- Teknik feature selection digunakan untuk memilih fitur-fitur mana sajakah yang diskriminatif.
- **Goal:** boosting model's performance, especially on very high dimensional datasets

Lihat http://scikit-learn.org/stable/modules/feature_selection.html

Univariate Feature Selection

Contoh: dari 28x28=784 fitur-fitur yang ada sebelumnya, kita akan coba lihat Top-100 fitur terbaik berdasarkan **chi-square statistical test**.

Selain chi-square, ada ANOVA, dsb.

http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest

```
from sklearn.datasets import load_svmlight_file
from sklearn import preprocessing
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

#load dataset in svmlib format
X, y = load_svmlight_file("dataset3.txt")

#X is scipy.sparse CSR matrix, we need to convert it to numpy array
X = X.toarray()

#scaling to [0,1]
min_max_scaler = preprocessing.MinMaxScaler()
X_scaled = min_max_scaler.fit_transform(X)

#using Chi-Square test to select top-100 features
X_new = SelectKBest(chi2, k=100).fit_transform(X_scaled, y)

print (X_new.shape)
print(X_new)
```

Data Tidak Seimbang ?

Unbalanced Data

- Terkadang kita menghadapi situasi dimana salah satu/beberapa kelas mempunyai proporsi yang jauh lebih besar dibandingkan kelas yang lainnya.

- Ini bisa menyebabkan model klasifikasi cenderung memberikan “preferensi lebih” kepada kelas yang proporsinya lebih banyak

Unbalanced Data

Teknik menghadapi masalah ini:

- Undersampling
  - Kurangi instances yang kelasnya dominan sehingga semuanya seimbang
- Oversampling
  - Instances yang kelasnya sedikit diperbesar hingga berukuran sama dengan instances kelas dominan (pilih secara random; pasti ada duplikasi)
- SMOTHE
  - Teknik oversampling/undersamplin yang lebih advanced, daripada hanya sekedar random

Unbalanced Data

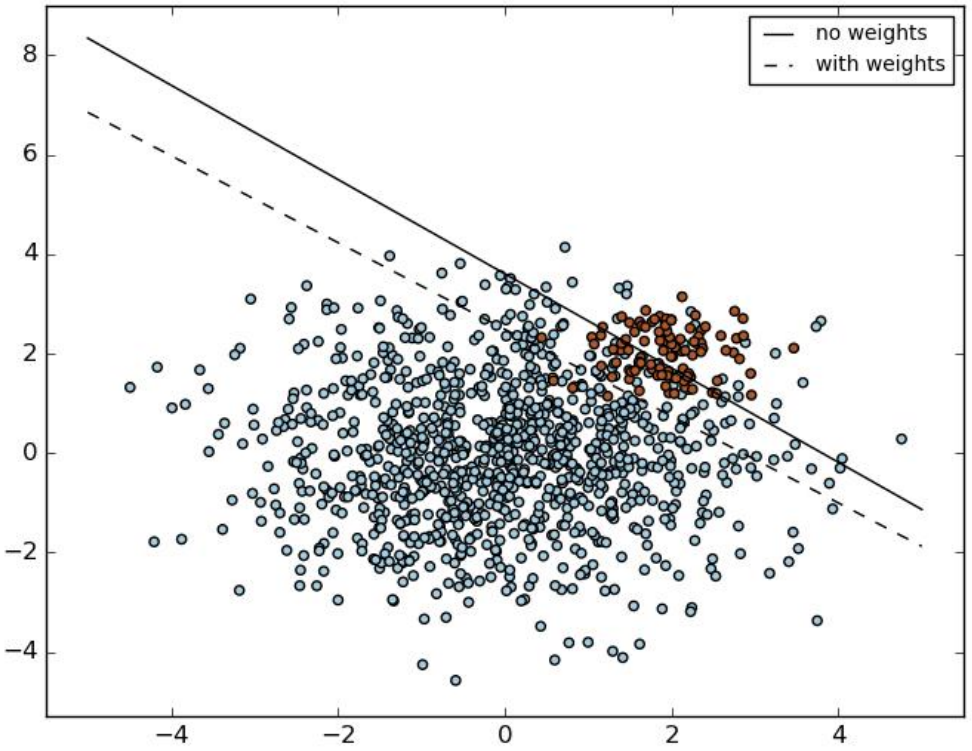
Teknik menghadapi masalah ini:

- Thresholding
  - Setiap model ML biasanya menghasilkan nilai probabilitas. Normalnya threshold yang digunakan adalah 0.5 untuk pembatas antara kelas (+) dengan kelas (-). Kita bisa mengganti nilai batas ini seandainya data kita tidak seimbang.

Unbalanced Data

Teknik menghadapi masalah ini:

- **Cost-sensitive ML model**: Salah satunya adalah SVM !
- Memberikan **bobot lebih** kepada instances yang labelnya sedikit



Lihat http://scikit-learn.org/stable/modules/svm.html

Automatic Model Selection

Model Selection

- **Goal:** mencari hyper-parameter yang dapat memberikan performa klasifikasi optimal (terhadap suatu metric tertentu)

- Hyper-parameter tersebut meliputi:
  - An estimator (classifier such as sklearn.svm.SVC())
  - A parameter space
  - A cross-validation scheme
  - A score function

Grid Search

The grid search exhaustively generates candidates from a grid of parameter values specified with the param_grid parameter. For instance, the following param_grid:

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
]
```

specifies that two grids should be explored: one with a linear kernel and C values in [1, 10, 100, 1000], and the second one with an RBF kernel, and the cross-product of C values ranging in [1, 10, 100, 1000] and gamma values in [0.001, 0.0001].

Detail http://scikit-learn.org/stable/modules/grid_search.html

Grid Search

- Pelajari kode gridsearch.py

Automatic Non-linear Feature Extraction

(Bernoulli Restricted Boltzmann Machine)

Non-linear Feature Extraction

- Bukankah fitur-fitur itu sudah ada? Yaitu ada 28x28=784 fitur yang bernilai 0 – 255 ??

- Ya betul, yang akan kita lakukan disini adalah kita ingin **mengekstraksi fitur-fitur yang lebih high-level lagi** !

Bernoulli Restricted Boltzmann Machine

- the Bernoulli Restricted Boltzmann machine model (BernoulliRBM) can perform effective non-linear feature extraction !

- Selain RBM, ada **Autoencoder**, dsb.

- Penjelasan RBM:

http://scikit-learn.org/stable/modules/neural_networks.html#rbm

Non-linear Feature Extraction

- Dengan dataset sebelumnya, kita akan coba extract fitur yang lebih high level dengan RBM, lalu setelah itu kita gunakan ML model yang linear seperti Logistic Regression untuk membangun model klasifikasi

- Pipeline: RBM -> Logistic Regression

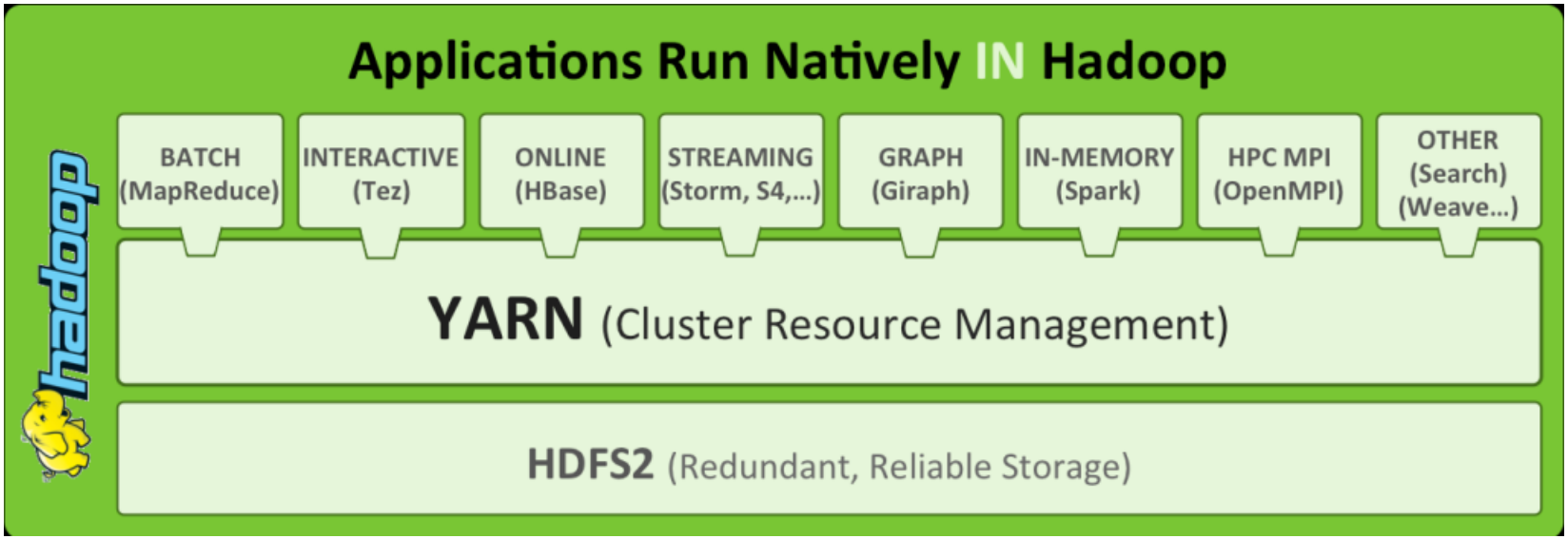
Non-linear Feature Extraction + Building Classification Model

- Code: lihat file rbm.py yang disediakan di tutorial ini

Jika Ukuran Dataset Besar?

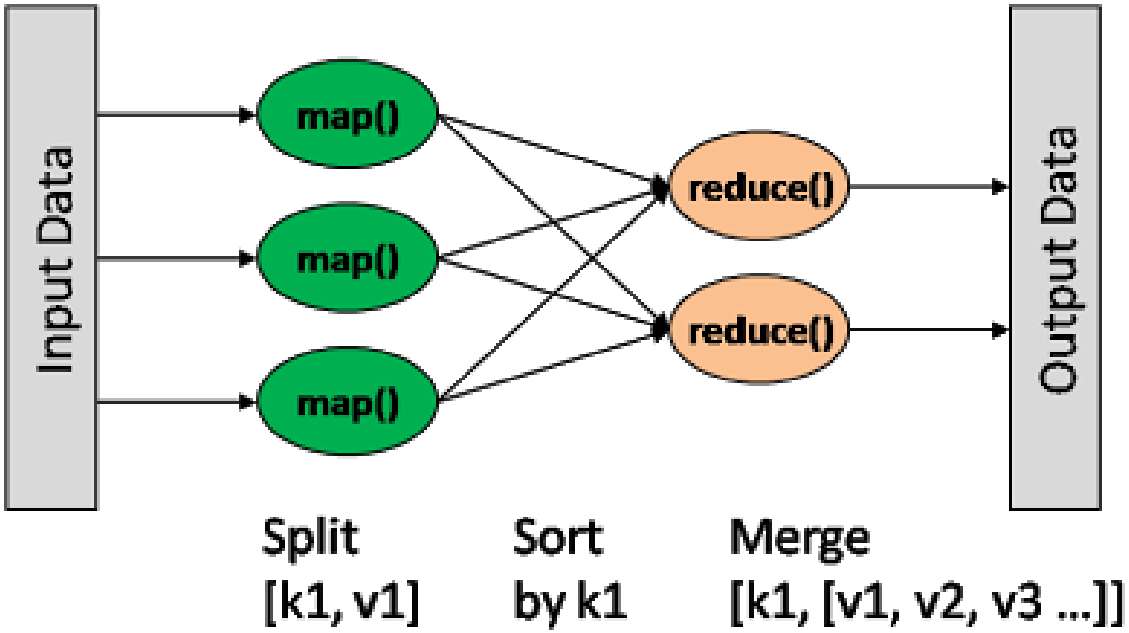
Platform untuk Big Data

Hadoop Ecosystem: Salah satu platform untuk pengolahan data besar !



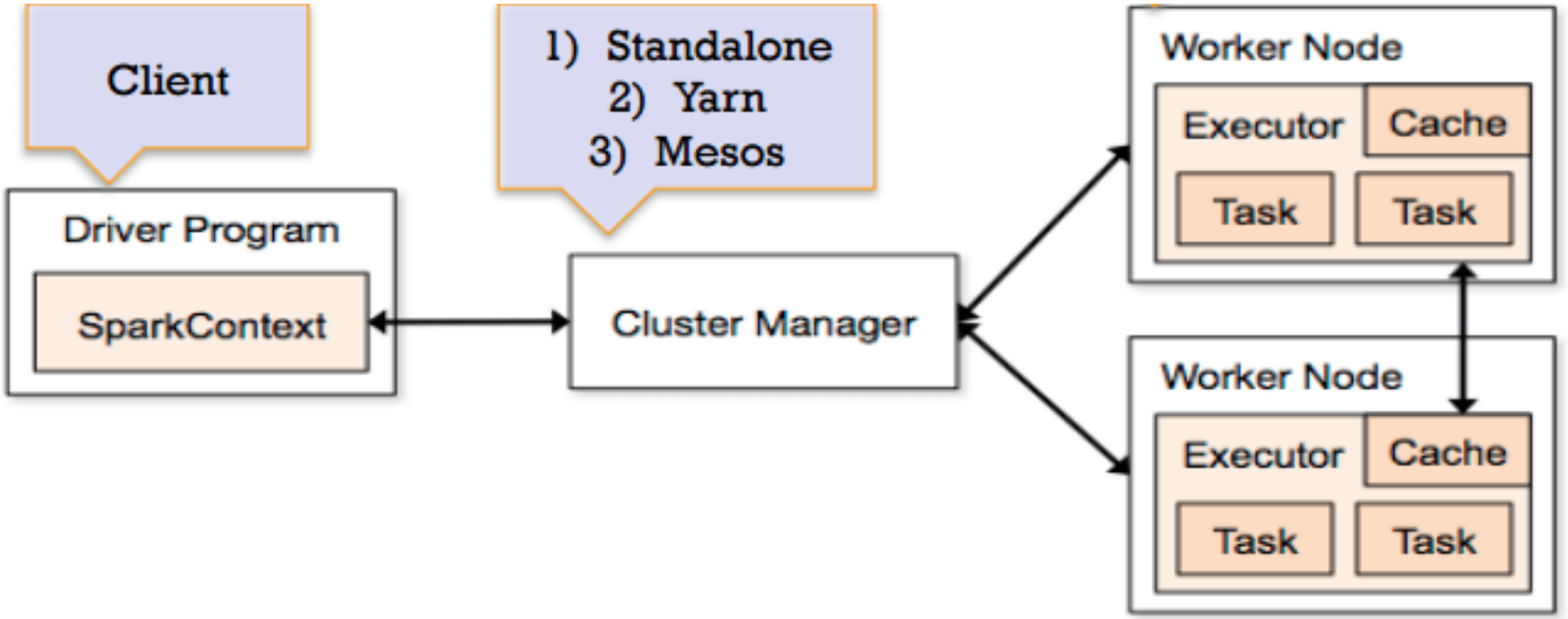
Map-Reduce

- Framework untuk distributed computing



https://dzone.com/articles/how-hadoop-mapreduce-works

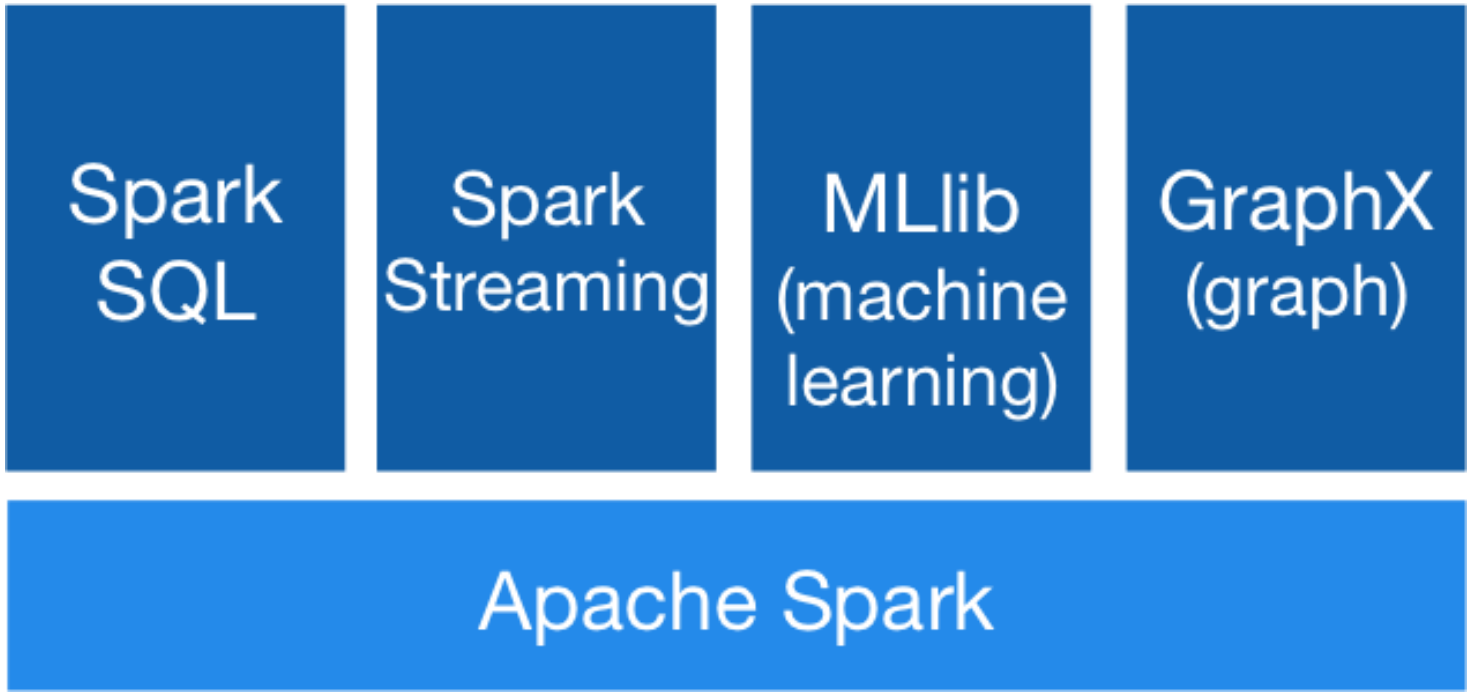
Spark



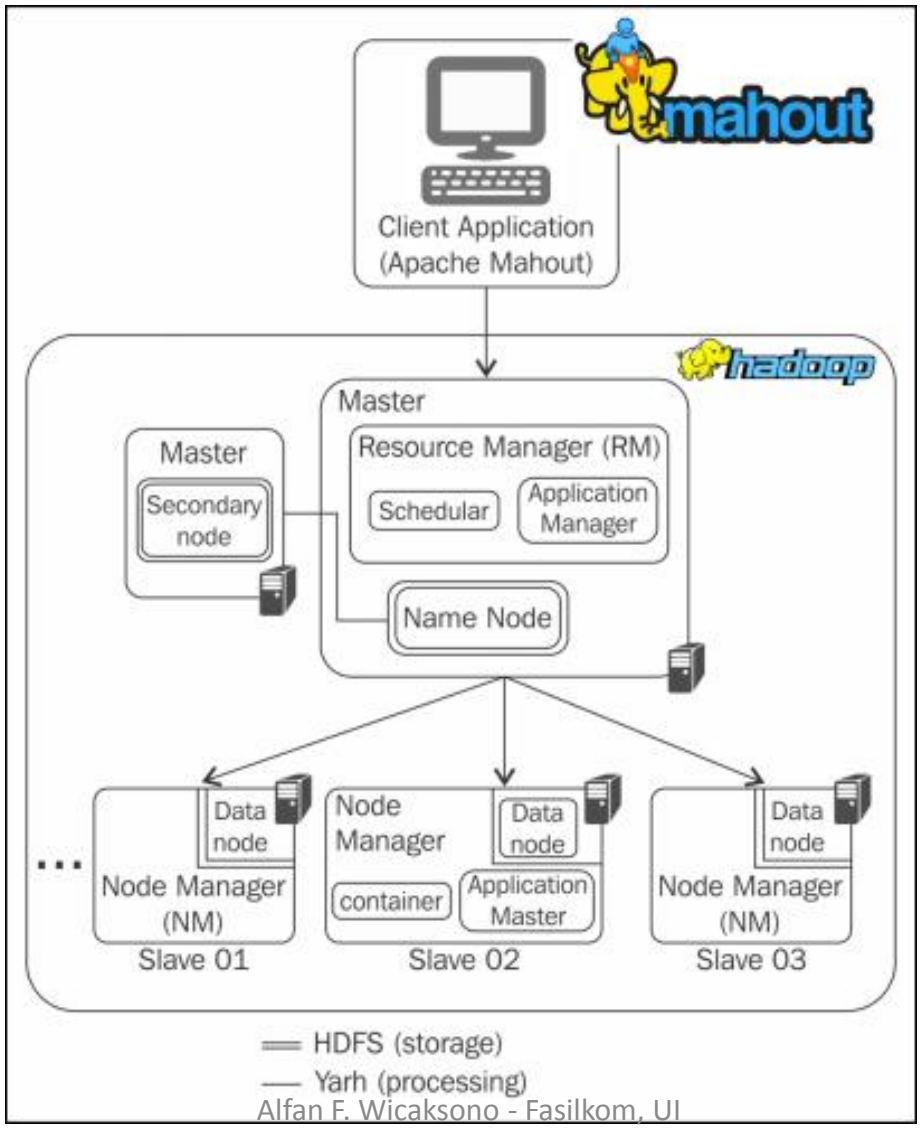
Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3.

http://jennyxiaozhang.com/6-things-you-need-to-know-about-spark/

Spark


Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Apache Spark

ML Tool: Mahout



ML Tool: Spark MLLib

- Sekarang, banyak orang menggunakan Spark untuk Machine Learning dibandingkan Mahout
- Alasan: komputasi lebih cepat, karena konsep In-Memory

Klasifikasi dengan SVM menggunakan Spark

```
from pyspark.mllib.classification import SVMWithSGD, SVMModel
from pyspark.mllib.regression import LabeledPoint

from pyspark import SparkContext
from pyspark.sql import SQLContext

sc = SparkContext(appName="svm")
sqlContext = SQLContext(sparkContext=sc)

# Load and parse the data
def parsePoint(line):
    values = [float(x) for x in line.split(' ')]
    return LabeledPoint(values[0], values[1:])

#ini lokasi file input di HDFS, relatif terhadap HOME DIR
data = sc.textFile("sample_svm_data.txt")
parsedData = data.map(parsePoint)
```

Klasifikasi dengan SVM menggunakan Spark

```
# Build the model
model = SVMWithSGD.train(parsedData, iterations=100)

# Evaluating the model on training data
labelsAndPreds = parsedData.map(lambda p: (p.label, \
                                           model.predict(p.features)))
trainErr = labelsAndPreds.filter(lambda (v, p): v != p)\
                                .count() / float(parsedData.count())
print("Training Error = " + str(trainErr))

# Save and load model
# direktori myModelPath ini akan dibuat di HDFSmodel.save(sc, "myModelPath")
sameModel = SVMModel.load(sc, "myModelPath")
```

Jika ingin install Hadoop Ecosystem

- Gunakan Distribusi Cloudera untuk kemudahan instalasi Hadoop beserta semua sub-sistemnya!
- http://www.cloudera.com/products.html