# Using decision trees in MLlib SPARK[¶](https://ocw.uc3m.es/ingenieria-informatica/machine-learning-i/spark_mllib.html#Using-decision-trees-in-MLlib-SPARK)

You can find how to use other algorithms (Random Forest, Gradient Boosting, etc.) [here](https://github.com/apache/spark/tree/master/examples/src/main/python/mllib)

In [1]:

import sys

import os

import os.path

SPARK\_HOME = """C:\spark-1.5.0-bin-hadoop2.6""" #CHANGE THIS PATH TO YOURS!

sys.path.append(os.path.join(SPARK\_HOME, "python", "lib", "py4j-0.8.2.1-src.zip"))

sys.path.append(os.path.join(SPARK\_HOME, "python", "lib", "pyspark.zip"))

os.environ["SPARK\_HOME"] = SPARK\_HOME

from pyspark import SparkContext

sc = SparkContext(master="local[\*]", appName="PythonDecisionTreeClassificationExample")

In [2]:

%matplotlib inline

from pyspark.mllib.regression import LabeledPoint

import numpy as np

import matplotlib.pyplot as plt

In [3]:

from sklearn.datasets import load\_iris

iris = load\_iris()

X = iris.data # Input attributes

y = iris.target # Label

# zip is used so that each instance is a tuble of (label, input attributes).

# This will make life easier later

# Note: zip([1,2,3], ["a","b","c"]) => [(1, 'a'), (2, 'b'), (3, 'c')]

data = zip(y,X)

In [4]:

data\_rdd = sc.parallelize(data,4)

print data\_rdd.getNumPartitions()

4

In [5]:

data\_rdd = data\_rdd.map(lambda x: LabeledPoint(x[0], x[1]))

data\_rdd.take(1)

Out[5]:

[LabeledPoint(0.0, [5.1,3.5,1.4,0.2])]

In [6]:

from pyspark.mllib.tree import DecisionTree, DecisionTreeModel

In [11]:

(trainingData\_rdd, testData\_rdd) = data\_rdd.randomSplit([0.7, 0.3])

In [13]:

model = DecisionTree.trainClassifier(trainingData\_rdd, numClasses=3, categoricalFeaturesInfo={},impurity='gini', maxDepth=5)

In [14]:

predictions = model.predict(testData\_rdd.map(lambda x: x.features))

labelsAndPredictions = testData\_rdd.map(lambda lp: lp.label).zip(predictions)

testErr = labelsAndPredictions.filter(lambda (v, p): v != p).count() / float(testData\_rdd.count())

print('Test Error = ' + str(testErr))

print('Learned classification tree model:')

print(model.toDebugString())

Test Error = 0.0208333333333

Learned classification tree model:

DecisionTreeModel classifier of depth 5 with 15 nodes

 If (feature 2 <= 1.7)

 Predict: 0.0

 Else (feature 2 > 1.7)

 If (feature 2 <= 4.8)

 If (feature 3 <= 1.6)

 Predict: 1.0

 Else (feature 3 > 1.6)

 If (feature 1 <= 2.8)

 Predict: 2.0

 Else (feature 1 > 2.8)

 Predict: 1.0

 Else (feature 2 > 4.8)

 If (feature 3 <= 1.7)

 If (feature 2 <= 5.0)

 If (feature 0 <= 6.0)

 Predict: 2.0

 Else (feature 0 > 6.0)

 Predict: 1.0

 Else (feature 2 > 5.0)

 Predict: 2.0

 Else (feature 3 > 1.7)

 Predict: 2.0

In [ ]:

