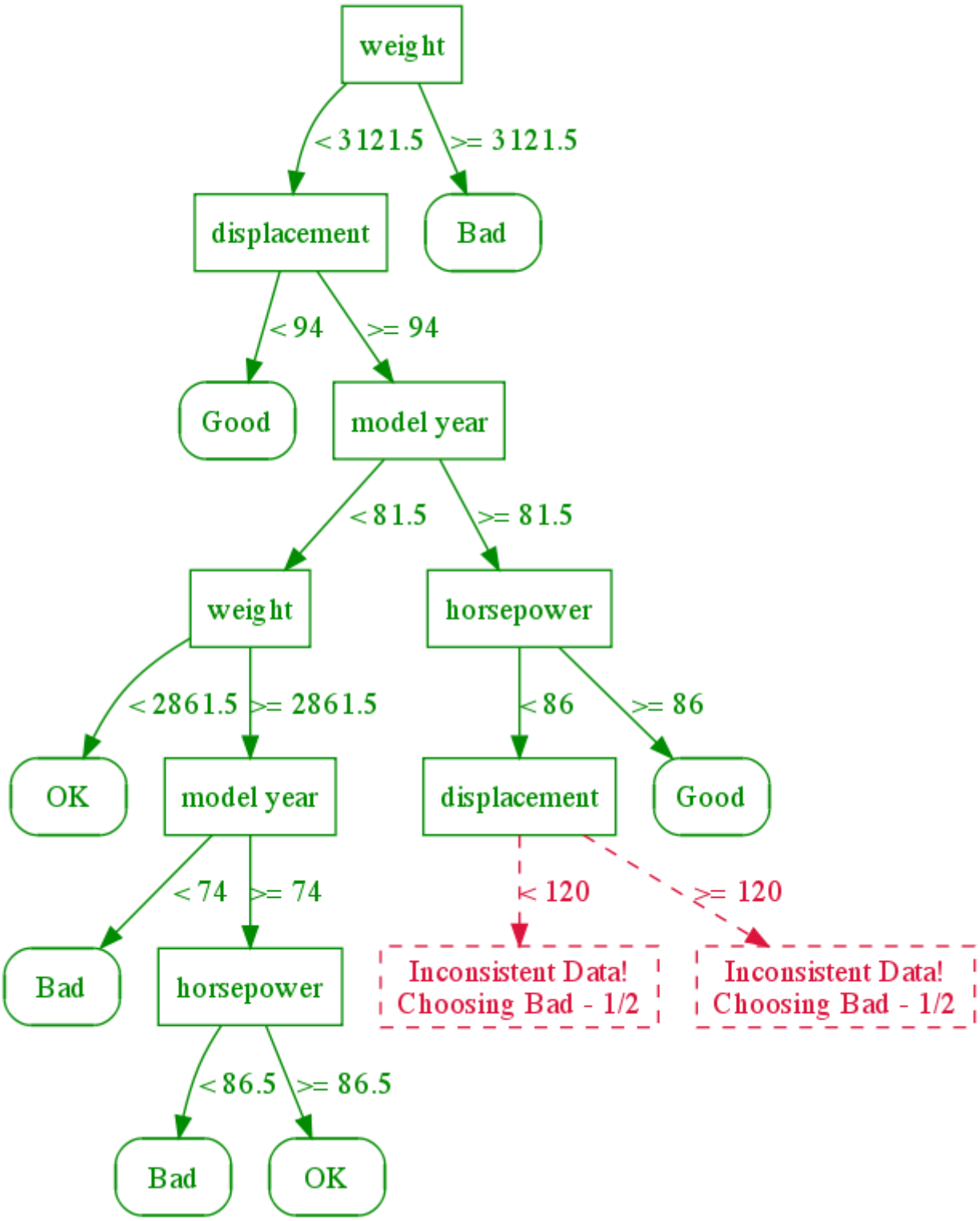
SPARK DECISION TREE CLASSIFIER

Here we explain how to use the Decision Tree Classifier with Apache Spark ML (machine learning). We use data from The University of Pennsylvania here and here. We write the solution in Scala code and walk the reader through each line of the code.

Do not bother to read the mathematics part of the lecture notes from Penn, unless you know a lot of math. Such knowledge is not necessary here. Instead this graphic below gives you all that you need to understand the basic idea:

The goal is to look at automobiles and predict gas mileage. The mileage is either bad (0), good (1), or ok (2) based on number of cylinders, engine displacement, horsepower, weight, acceleration time to 60 mph, model year, and continent of origin. Below the graph is the data in spreadsheet format with text, and below that, the text is converted to numbers.

The graph below describes how the decision tree works. First you look at the weight. If it is heavy the mileage is bad. If it is not heavy and the engine displacement is high then you have to look at model year. You keep dividing this problem until you get to a point where logically it makes no sense to go any further. Obviously if a car weighs a whole lot, then rightaway you can say that its mileage will be bad.



Graphic Source: University of Pennsylvania

Here is the data with column headings.

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
Bad	8	350	150	4699	14.5	74	America
Bad	8	400	170	4746	12	71	America
Bad	8	400	175	4385	12	72	America
Bad	6	250	72	3158	19.5	75	America

OK	4	121	98	2945	14.5	75	Asia
OK	6	232	90	3085	17.6	76	America
OK	4	120	97	2506	14.5	72	Europe

And here it is converted to numeric format so we can use it with the ML algorithm. Download that data from here and save it to your drive or put it in Hadoop.

0	8	350	150	4699	14.5	74	0
0	8	400	170	4746	12	71	0
0	8	400	175	4385	12	72	0
0	6	250	72	3158	19.5	75	0
0	8	304	150	3892	12.5	72	0
0	8	350	145	4440	14	75	0
0	6	250	105	3897	18.5	75	0
0	6	163	133	3410	15.8	78	1
0	8	260	110	4060	19	77	0
1	4	97	75	2171	16	75	2
1	4	97	78	1940	14.5	77	1
1	4	98	83	2219	16.5	74	1
2	4	79	70	2074	19.5	71	1
2	4	91	68	1970	17.6	82	2
2	4	89	71	1925	14	79	1
2	4	83	61	2003	19	74	2

Scala Code

Run spark-shell or create a Zeppelin notebook and paste in the code below.

```
import org.apache.spark.mllib.tree.DecisionTree
import org.apache.spark.mllib.tree.model.DecisionTreeModel
import org.apache.spark.mllib.util.MLUtils
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.feature.Normalizer
// read the data file and split it by spaces
val data = sc.textFile("/home/walker/mileage.txt").map( l => l.split("\\s+"))
// convert every string to doubles
var dbl = data.map (m => m.map(l => l.toDouble))
// Here we create a Dense Vector and LabeledPoint. A Dense Vector is a
// special kind of Vector than the scala regular Vector. So use Vectors, with
an “s”
// at the end. The values must be doubles. A Dense Vector cannot handle
// blank values. If we had that those, we would use a Sparse Vector.
//
// The LabeledPoint is an object with the label and features. The label
// is 0, 1, 2 or gas mileage and the features are the weight, year, etc.
val parsedData = dbl.map { y =>
val c = Vectors.dense(y(1),y(2),y(3),y(4),y(5),y(6),y(7))
LabeledPoint(y(0), c)
}.cache()
```

```
// numClasses means number of classifications. In this case it is
// 0, 1, or 2
val numClasses = 3
// Here we say that the 6th feature is categorical with 3 possible values.
// It can be American, European, or Asian.
// The rest of the values are continuous, meaning just numbers and not
discrete
// values.
val categoricalFeaturesInfo = Map(6 -> 3)
val impurity = "gini"
val maxDepth = 9
val maxBins = 7
// Now feed the data into the model.
val model = DecisionTree.trainClassifier(parsedData, numClasses,
categoricalFeaturesInfo , impurity, maxDepth, maxBins)
// Print out the results
println("Learned classification tree model:\n" + model.toDebugString)
```

Results:
DecisionTreeModel classifier of depth 5 with 21 nodes

Here it says that we have 21 nodes (bottom endpoints) and 5 levels of decisions to make. For example if feature 3 (weight) is <= 2930 then we have to look at feature 1 (displacement and so on) to derive our prediction of 0, 1, 2.

```
If (feature 3 <= 2930.0)
If (feature 1 <= 91.0)
Predict: 2.0
Else (feature 1 > 91.0)
If (feature 5 <= 81.0)
If (feature 2 <= 105.0)
Predict: 1.0
Else (feature 2 > 105.0)
If (feature 0 <= 4.0)
Predict: 0.0
Else (feature 0 > 4.0)
Predict: 1.0
Else (feature 5 > 81.0)
If (feature 2 <= 85.0)
Predict: 0.0
Else (feature 2 > 85.0)
Predict: 2.0
Else (feature 3 > 2930.0)
If (feature 3 <= 3193.0)
If (feature 2 <= 85.0)
Predict: 0.0
Else (feature 2 > 85.0)
If (feature 2 <= 105.0)
```

```
If (feature 0 <= 4.0)
Predict: 1.0
Else (feature 0 > 4.0)
Predict: 0.0
Else (feature 2 > 105.0)
Predict: 0.0
Else (feature 3 > 3193.0)
Predict: 0.0
```

We would then save this model and use the predict() method to pass in one particular vehicle to predict what kind of mileage it will have.

You might have noticed that this kind of model is similar to logistic regression. But we use decision trees (there are several types) when we want to handle much data as it is less computationally expensive. In other words it runs faster and consumes less memory.

Also note that the Penn lecture used the matlab math and stats programming tool. But those kinds of tools cannot work with big data. They do not scale like Apache Spark to millions of decisions or more.