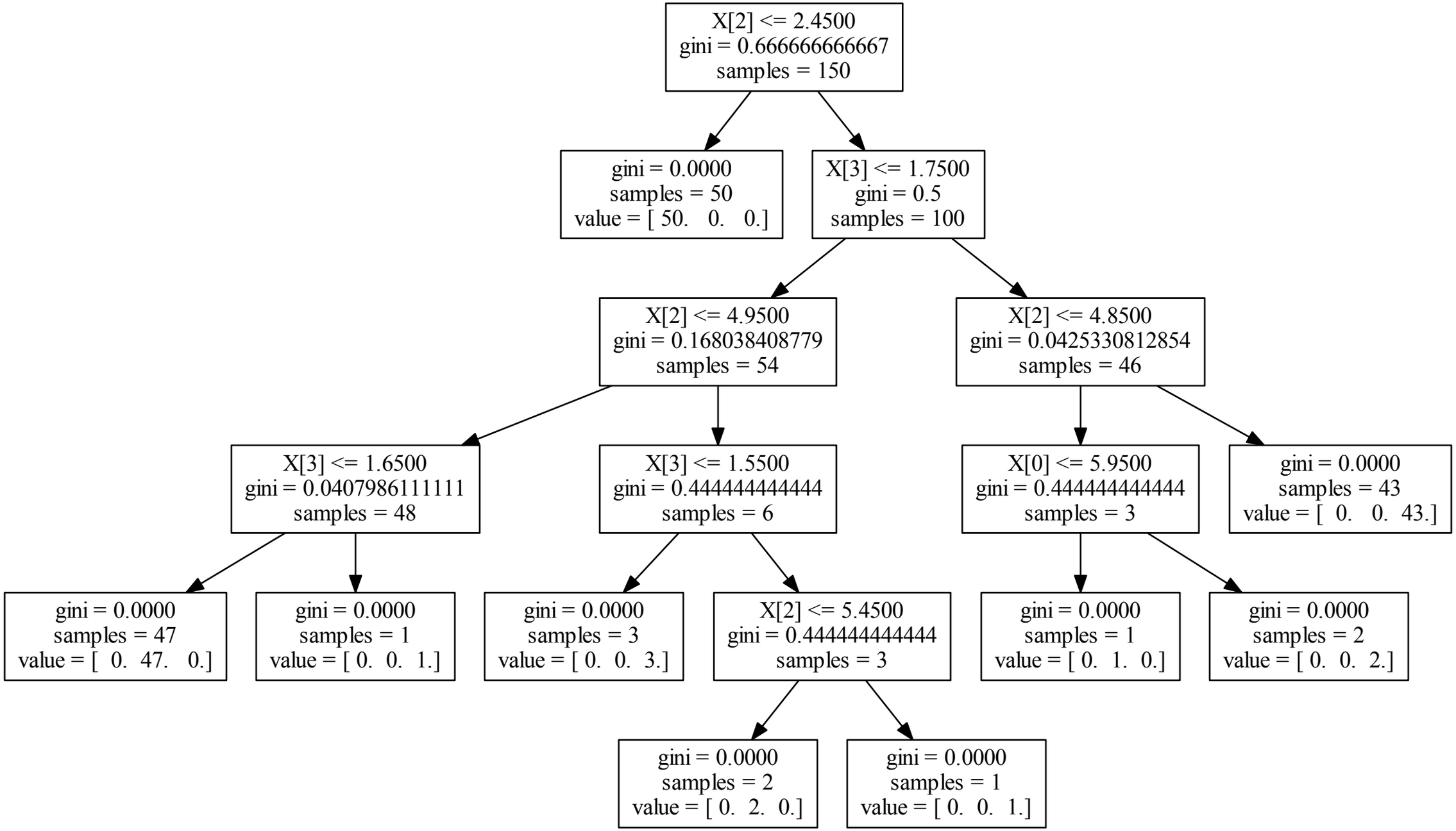
Topic_6

April 4, 2015

In [ ]:
```
#Author: Muhammed Khan
from sklearn.datasets import load_iris
from sklearn import tree #import decision tree library for classifiction
from sklearn.externals.six import StringIO  #read and writes strings to memory
# import dot_parser
import pydot #python interface for graphviz's dot language

iris = load_iris()
clf = tree.DecisionTreeClassifier() #load classifier from scikit-learn
clf = clf.fit(iris.data, iris.target)  #learning from our existing data, use
#classifier to fit data to target attribute by
#building an estimator

from sklearn.externals.six import StringIO
with open("iris.dot", 'w') as f:
    f = tree.export_graphviz(clf, out_file=f) #export decision tree from clf
    #estimates into graphviz file f

dot_data = StringIO() #create StringIO() object to store dot_data
tree.export_graphviz(clf, out_file=dot_data) #export tree to Graphviz dot.data
#file
graph = pydot.graph_from_dot_data(dot_data.getvalue()) #access pydot and get
#values from dota_data
graph.write_pdf("iris.pdf") #write graph into pdf
```

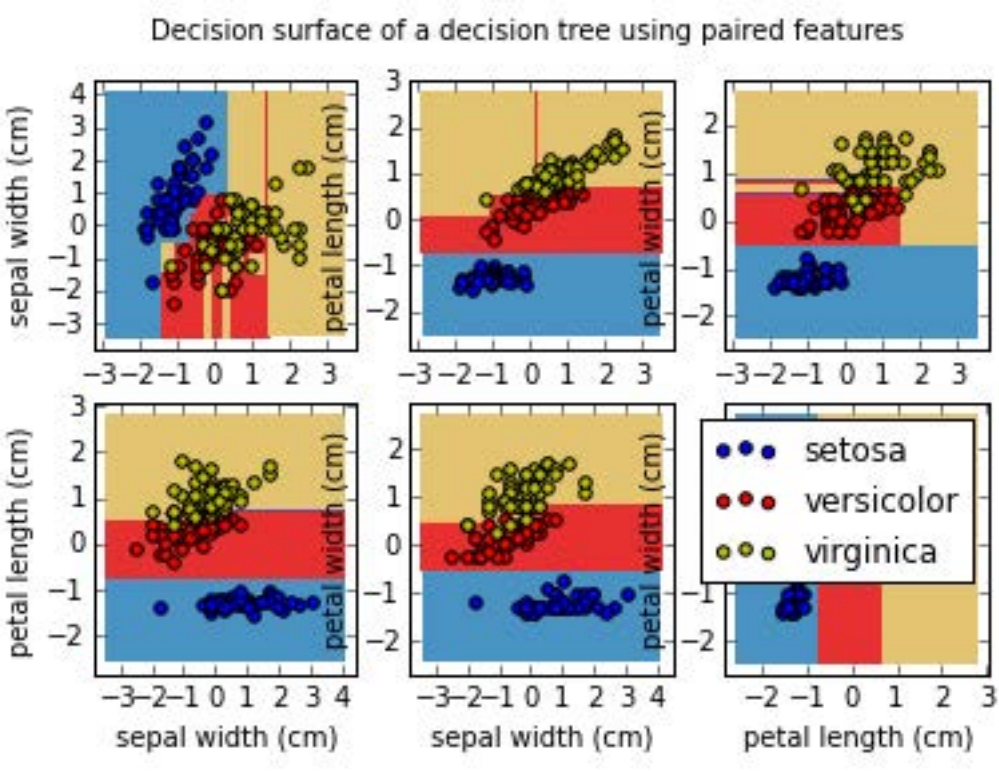

X[2] <= 2.4500
gini = 0.666666666667
samples = 150
gini = 0.0000
samples = 50
value = [50. 0. 0.]
X[3] <= 1.7500
gini = 0.5
samples = 100
X[2] <= 4.9500
gini = 0.168038408779
samples = 54
X[2] <= 4.8500
gini = 0.0425330812854
samples = 46
X[3] <= 1.6500
gini = 0.0407986111111
samples = 48
X[3] <= 1.5500
gini = 0.444444444444
samples = 6
X[0] <= 5.9500
gini = 0.444444444444
samples = 3
gini = 0.0000
samples = 43
value = [0. 0. 43.]
gini = 0.0000
samples = 47
value = [0. 47. 0.]
gini = 0.0000
samples = 1
value = [0. 0. 1.]
gini = 0.0000
samples = 3
value = [0. 0. 3.]
X[2] <= 5.4500
gini = 0.444444444444
samples = 3
gini = 0.0000
samples = 1
value = [0. 1. 0.]
gini = 0.0000
samples = 2
value = [0. 0. 2.]
gini = 0.0000
samples = 2
value = [0. 2. 0.]
gini = 0.0000
samples = 1
value = [0. 0. 1.]

plot_iris

April 4, 2015

In [1]:
```
#author: Muhammed Khan

#Source: http://scikit-learn.org/stable/auto_examples/tree/plot_iris.html

#Plots the decision surface of a decision tree based on ALL features,
#not just the target features of the dataset
print(__doc__)

import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

# Parameters
n_classes = 3
plot_colors = "bry"
plot_step = 0.02

# Load data
iris = load_iris()

for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3],
                                [1, 2], [1, 3], [2, 3]]):
    # We only take the two corresponding features
    X = iris.data[:, pair]
    y = iris.target

    # Shuffle
    idx = np.arange(X.shape[0])
    np.random.seed(13)
    np.random.shuffle(idx)
    X = X[idx]
    y = y[idx]

    # Standardize
    mean = X.mean(axis=0)
    std = X.std(axis=0)
    X = (X - mean) / std

    # Train
    clf = DecisionTreeClassifier().fit(X, y)
```

```
            # Plot the decision boundary
            plt.subplot(2, 3, pairidx + 1)

            x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
            y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
            xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                                 np.arange(y_min, y_max, plot_step))

            Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
            Z = Z.reshape(xx.shape)
            cs = plt.contourf(xx, yy, Z, cmap=plt.cm.Paired)

            plt.xlabel(iris.feature_names[pair[0]])
            plt.ylabel(iris.feature_names[pair[1]])
            plt.axis("tight")

            # Plot the training points
            for i, color in zip(range(n_classes), plot_colors):
                idx = np.where(y == i)
                plt.scatter(X[idx, 0], X[idx, 1], c=color, label=iris.target_names[i],
                            cmap=plt.cm.Paired)

            plt.axis("tight")

        plt.suptitle("Decision surface of a decision tree using paired features")
        plt.legend()
        plt.show()

Automatically created module for IPython interactive environment
```

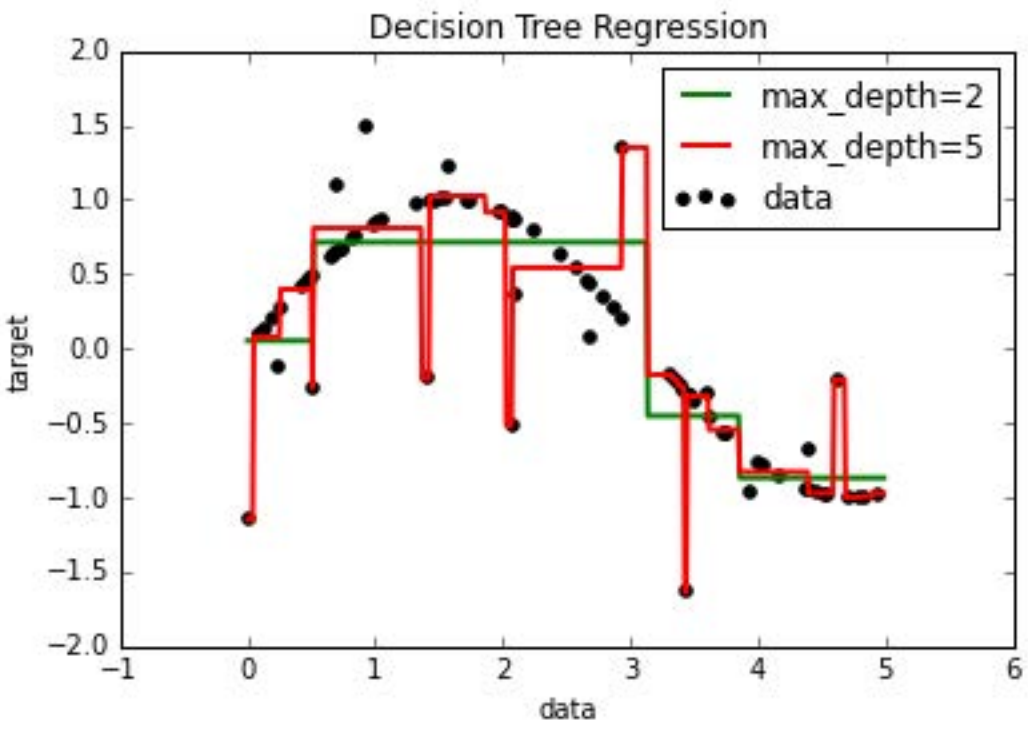

Decision surface of a decision tree using paired features
sepal width (cm)
petal length (cm)
petal width (cm)
petal length (cm)
petal width (cm)
petal length (cm)
sepal width (cm)
sepal width (cm)
petal length (cm)
setosa
versicolor
virginica

plot_tree_regression

April 4, 2015

In [4]:
```
#author: Muhammed Khan
#Source:http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html
#Use sklearn.trees to approximate sine curve to noisy data
print(__doc__)

# Import the necessary modules and libraries
import numpy as np
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt

# Create a random dataset
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0) #calling random number generator on
#axis
y = np.sin(X).ravel() #returns a 1D array of X
y[::5] += 3 * (0.5 - rng.rand(16))

# Fit regression model
clf_1 = DecisionTreeRegressor(max_depth=2) #underfitting noise
clf_2 = DecisionTreeRegressor(max_depth=5) #overfitting noise
clf_1.fit(X, y)
clf_2.fit(X, y)

# Predict
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]
y_1 = clf_1.predict(X_test)
y_2 = clf_2.predict(X_test)

# Plot the results
plt.figure()
plt.scatter(X, y, c="k", label="data")
plt.plot(X_test, y_1, c="g", label="max_depth=2", linewidth=2)
plt.plot(X_test, y_2, c="r", label="max_depth=5", linewidth=2)
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```

```
Automatically created module for IPython interactive environment
```


Decision Tree Regression
max_depth=2
max_depth=5
data
target
data
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1
0
1
2
3
4
5
6