DecisionTreesExplained

January 18, 2018

1 Decision Trees Explained

Decision trees seem hard to understand but they are just if-then-else rules. The conditionals are chosen by the data. Thus, we rarely use a single decision tree as it will probably have poor generalization.

http://stackoverflow.com/questions/20224526/how-to-extract-the-decision-rules-from-scikit-learn-decision-tree

Here we try to explain a decision tree using a simple example.

```
In [21]: import pandas as pd
         import numpy as np
         from sklearn.tree import DecisionTreeClassifier
         from sklearn import tree
         from sklearn.externals.six import StringIO as StringIO
         import pydot

         # dummy data:
         df = pd.DataFrame({'col1':[0,1,2,3],'col2':[3,4,5,6],'dv':[0,1,0,1]})

         # create decision tree
         dt = DecisionTreeClassifier(max_depth=5, min_samples_leaf=1)
         dt.fit(df.ix[:,:2], df.dv)
```

```
Out[21]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
                     max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
                     min_samples_split=2, min_weight_fraction_leaf=0.0,
                     presort=False, random_state=None, splitter='best')
```

```
In [2]: df.ix[:,:2].columns
```

```
Out[2]: Index(['col1', 'col2'], dtype='object')
```

```
In [26]: def print_decision_tree(tree, feature_names=None, offset_unit='    '):
             '''
             Plots textual representation of rules of a decision tree

             tree: scikit-learn representation of tree
```

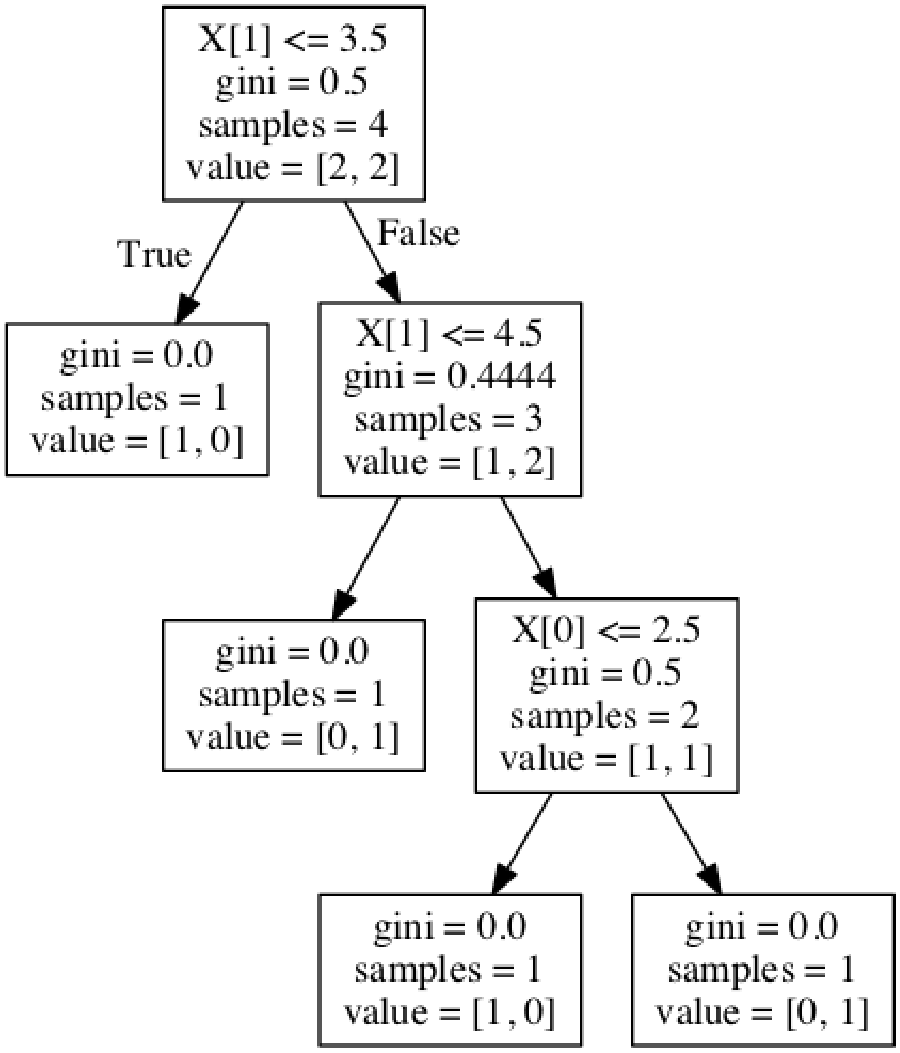
```
            feature_names: list of feature names. They are set to f1,f2,f3,... if not specified
            offset_unit: a string of offset of the conditional block

            See http://stackoverflow.com/a/35840109
            '''
            left      = tree.tree_.children_left
            right     = tree.tree_.children_right
            threshold = tree.tree_.threshold
            value = tree.tree_.value
            if feature_names is None:
                features  = ['f%d'.format(i) for i in tree.tree_.feature]
            else:
                features  = [feature_names[i] for i in tree.tree_.feature]

            def recurse(left, right, threshold, features, node, depth=0):
                    offset = offset_unit*depth
                    if (threshold[node] != -2):
                            print(offset+"if ( " + features[node] + " <= " + str(threshold[node
                            if left[node] != -1:
                                    recurse (left, right, threshold, features,left[node],depth+
                            print(offset+"} else {")
                            if right[node] != -1:
                                    recurse (left, right, threshold, features,right[node],depth
                            print(offset+"}")
                    else:
                            print(offset+"return " + str(value[node]))

            recurse(left, right, threshold, features, 0,0)
```

```
In [27]: print_decision_tree(dt, df.ix[:,:2].columns)
```

```
if ( col2 <= 3.5 ) {
    return [[ 1.  0.]]
} else {
    if ( col2 <= 4.5 ) {
        return [[ 0.  1.]]
    } else {
        if ( col1 <= 2.5 ) {
            return [[ 1.  0.]]
        } else {
            return [[ 0.  1.]]
        }
    }
}
```

```
In [25]: with open('tree.dot', 'w') as f:
             f = tree.export_graphviz(dt, out_file=f)
```

Now we can run the command `dot -Tpng tree.dot -o tree.png` at the command line.


X[1] <= 3.5
gini = 0.5
samples = 4
value = [2, 2]
True
False
gini = 0.0
samples = 1
value = [1, 0]
X[1] <= 4.5
gini = 0.4444
samples = 3
value = [1, 2]
gini = 0.0
samples = 1
value = [0, 1]
X[0] <= 2.5
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [0, 1]

tree