DecisionTree

December 20, 2016

1 Decison Trees

First we'll load some fake data on past hires I made up. Note how we use pandas to convert a csv file into a DataFrame:

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn import tree

        input_file = "e:/sundog-consult/udemy/datascience/PastHires.csv"
        df = pd.read_csv(input_file, header = 0)
```

```
In [2]: df.head()
```

```
Out[2]:    Years Experience Employed?  Previous employers Level of Education  \
        0                10         Y                   4                 BS
        1                 0         N                   0                 BS
        2                 7         N                   6                 BS
        3                 2         Y                   1                 MS
        4                20         N                   2                PhD

          Top-tier school Interned Hired
        0               N        N     Y
        1               Y        Y     Y
        2               N        N     N
        3               Y        N     Y
        4               Y        N     N
```

scikit-learn needs everything to be numerical for decision trees to work. So, we'll map Y,N to 1,0 and levels of education to some scale of 0-2. In the real world, you'd need to think about how to deal with unexpected or missing data! By using map(), we know we'll get NaN for unexpected values.

```
In [3]: d = {'Y': 1, 'N': 0}
        df['Hired'] = df['Hired'].map(d)
        df['Employed?'] = df['Employed?'].map(d)
        df['Top-tier school'] = df['Top-tier school'].map(d)
        df['Interned'] = df['Interned'].map(d)
```

```
d = {'BS': 0, 'MS': 1, 'PhD': 2}
df['Level of Education'] = df['Level of Education'].map(d)
df.head()
```

Out[3]:

	Years Experience	Employed?	Previous employers	Level of Education	Top-tier school	Interned	Hired
0	10	1	4	0	0	0	1
1	0	0	0	0	1	1	1
2	7	0	6	0	0	0	0
3	2	1	1	1	1	0	1
4	20	0	2	2	1	0	0

Next we need to separate the features from the target column that we're trying to bulid a decision tree for.

```
In [4]: features = list(df.columns[:6])
        features
```

```
Out[4]: ['Years Experience',
         'Employed?',
         'Previous employers',
         'Level of Education',
         'Top-tier school',
         'Interned']
```

Now actually construct the decision tree:

```
In [5]: y = df["Hired"]
        X = df[features]
        clf = tree.DecisionTreeClassifier()
        clf = clf.fit(X,y)
```

... and display it. Note you need to have pyplot2 installed for this to work.

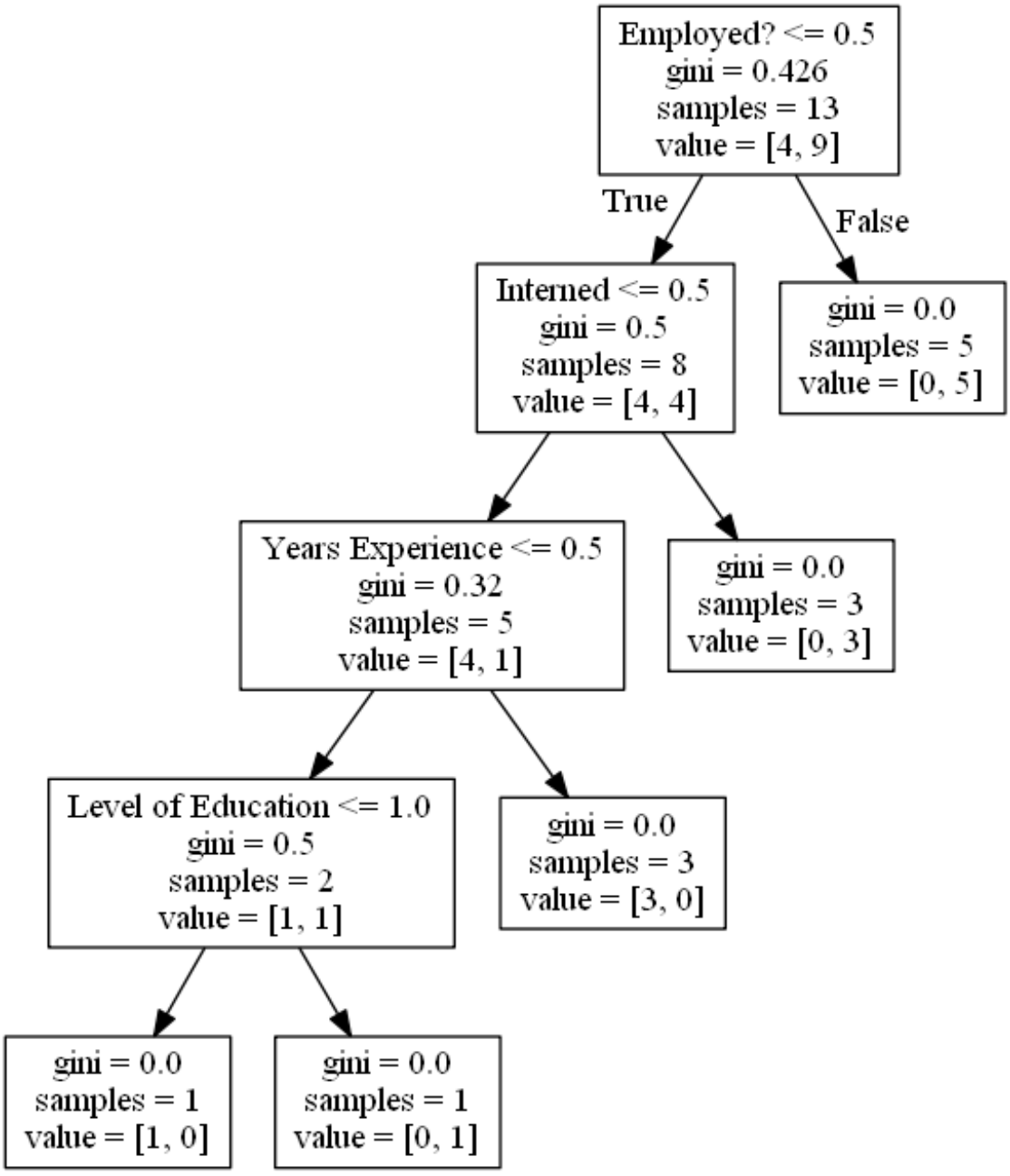
To read this decision tree, each condition branches left for "true" and right for "false". When you end up at a value, the value array represents how many samples exist in each target value. So value = [0. 5.] mean there are 0 "no hires" and 5 "hires" by the tim we get to that point. value = [3. 0.] means 3 no-hires and 0 hires.

```
In [6]: from IPython.display import Image
        from sklearn.externals.six import StringIO
        import pydot

        dot_data = StringIO()
```

```
tree.export_graphviz(clf, out_file=dot_data,
                                feature_names=features)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[6]:



1.1 Ensemble learning: using a random forest

We'll use a random forest of 10 decision trees to predict employment of specific candidate profiles:

```
In [8]: from sklearn.ensemble import RandomForestClassifier

        clf = RandomForestClassifier(n_estimators=10)
        clf = clf.fit(X, y)

        #Predict employment of an employed 10-year veteran
        print clf.predict([[10, 1, 4, 0, 0, 0]])
        #...and an unemployed 10-year veteran
        print clf.predict([[10, 0, 4, 0, 0, 0]])
```

```
[1]
[0]
```

1.2 Activity

Modify the test data to create an alternate universe where everyone I hire everyone I normally wouldn't have, and vice versa. Compare the resulting decision tree to the one from the original data.

```
In [ ]:
```