Decision Trees and Random Forests Project1

October 26, 2016

1 Decision Trees and Random Forests in Python

1.1 Import Libraries

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
```

1.2 Get the Data

```
In [3]: df = pd.read_csv(’kyphosis.csv’)
```

```
In [4]: df.info()
```

```
<class ’pandas.core.frame.DataFrame’>
RangeIndex: 81 entries, 0 to 80
Data columns (total 4 columns):
Kyphosis    81 non-null object
Age         81 non-null int64
Number      81 non-null int64
Start       81 non-null int64
dtypes: int64(3), object(1)
memory usage: 2.6+ KB
```

```
In [21]: df.head()
```

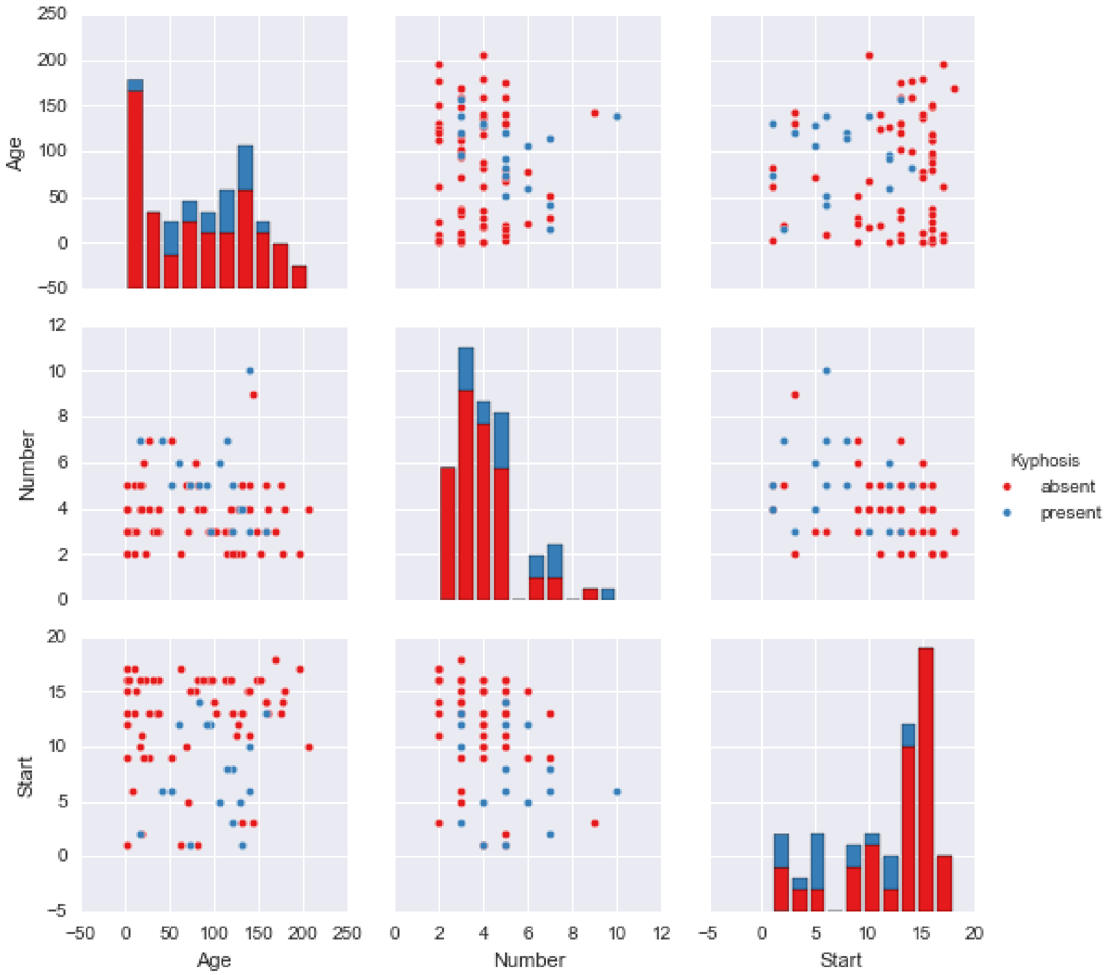
```
Out[21]:   Kyphosis  Age  Number  Start
         0   absent   71       3      5
         1   absent  158       3     14
         2  present  128       4      5
         3   absent    2       5      1
         4   absent    1       4     15
```

1.3 EDA

We’ll just check out a simple pairplot for this small dataset.

```
In [27]: sns.pairplot(df,hue=’Kyphosis’,palette=’Set1’)
```

```
Out[27]: <seaborn.axisgrid.PairGrid at 0x11b285f28>
```



1.4 Train Test Split

Let's split up the data into a training set and a test set!

```
In [13]: from sklearn.cross_validation import train_test_split
```

```
In [14]: X = df.drop('Kyphosis',axis=1)
         y = df['Kyphosis']
```

```
In [15]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

1.5 Decision Trees

We'll start just by training a single decision tree.

```
In [10]: from sklearn.tree import DecisionTreeClassifier
```

```
In [11]: dtree = DecisionTreeClassifier()
```

```
In [16]: dtree.fit(X_train,y_train)
```

```
Out[16]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                     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')
```

1.6 Prediction and Evaluation

Let's evaluate our decision tree.

```
In [17]: predictions = dtree.predict(X_test)
```

```
In [18]: from sklearn.metrics import classification_report,confusion_matrix
```

```
In [19]: print(classification_report(y_test,predictions))
```

```
precision    recall  f1-score   support

     absent       0.85      0.85      0.85        20
    present       0.40      0.40      0.40         5

avg / total       0.76      0.76      0.76        25
```

```
In [20]: print(confusion_matrix(y_test,predictions))
```

```
[[17  3]
 [ 3  2]]
```

1.7 Tree Visualization

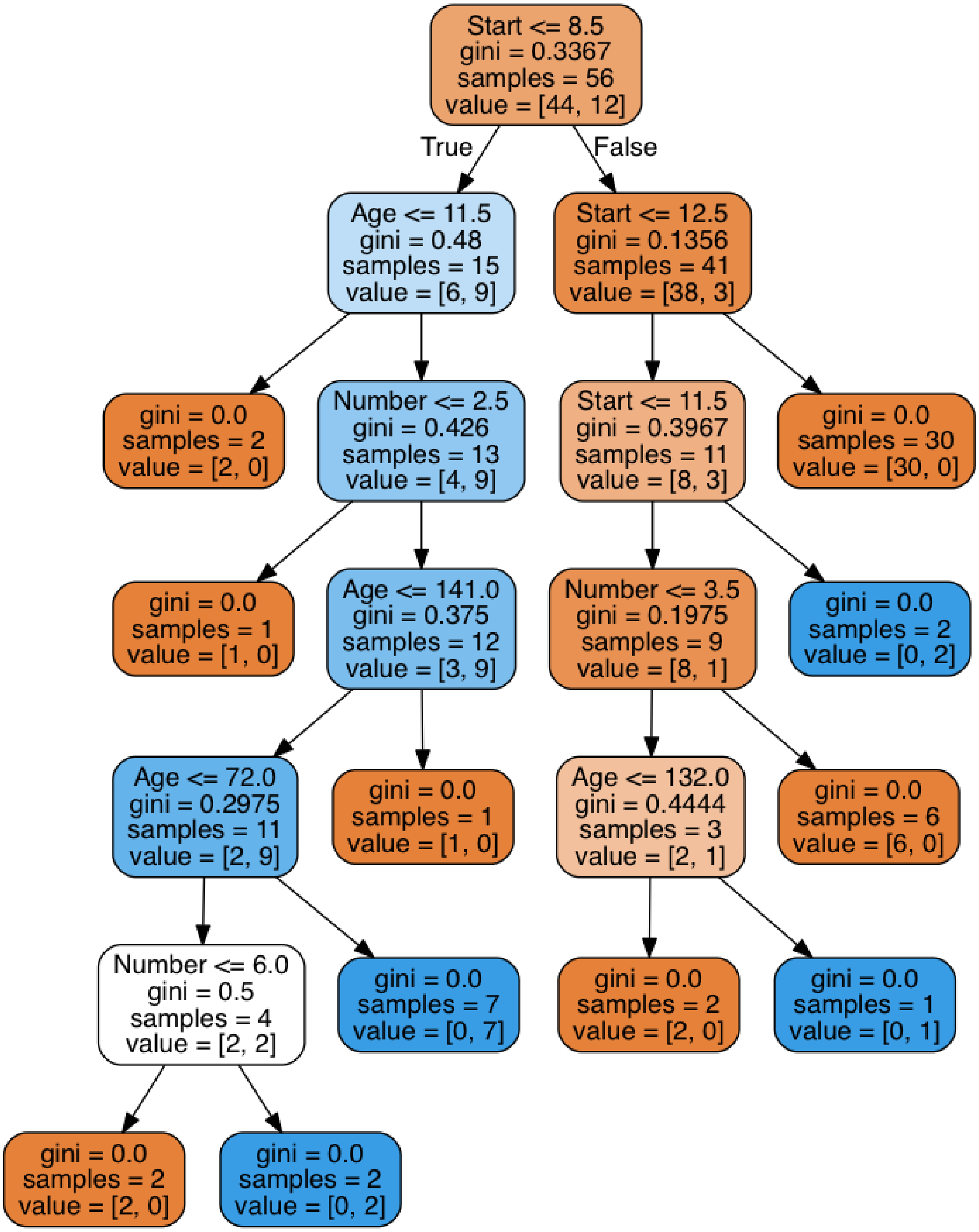
Scikit learn actually has some built-in visualization capabilities for decision trees, you won't use this often and it requires you to install the pydot library, but here is an example of what it looks like and the code to execute this:

```
In [33]: from IPython.display import Image
         from sklearn.externals.six import StringIO
         from sklearn.tree import export_graphviz
         import pydot

         features = list(df.columns[1:])
         features
```

```
Out[33]: ['Age', 'Number', 'Start']
```

```
In [39]: dot_data = StringIO()
         export_graphviz(dtree, out_file=dot_data,feature_names=features,filled=True,rounded=True)

         graph = pydot.graph_from_dot_data(dot_data.getvalue())
         Image(graph[0].create_png())
```

```
Out[39]:
```



1.8 Random Forests

Now let's compare the decision tree model to a random forest.

```
In [41]: from sklearn.ensemble import RandomForestClassifier
         rfc = RandomForestClassifier(n_estimators=100)
         rfc.fit(X_train, y_train)
```

```
Out[41]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False)
```

```
In [45]: rfc_pred = rfc.predict(X_test)
```

```
In [46]: print(confusion_matrix(y_test,rfc_pred))
```

```
[[18  2]
 [ 3  2]]
```

```
In [47]: print(classification_report(y_test,rfc_pred))
```

```
precision    recall  f1-score   support

     absent       0.86      0.90      0.88        20
    present       0.50      0.40      0.44         5

avg / total       0.79      0.80      0.79        25
```

2 Great Job!