Árvores de decisão e florestas aleatórias em Python

1 Árvores de decisão e florestas aleatórias em Python

1.1 Importando bibliotecas

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

1.2 Obtendo os dados

[2]:
```
df = pd.read_csv('kyphosis.csv')
```

[3]:
```
df.head()
```

[3]:

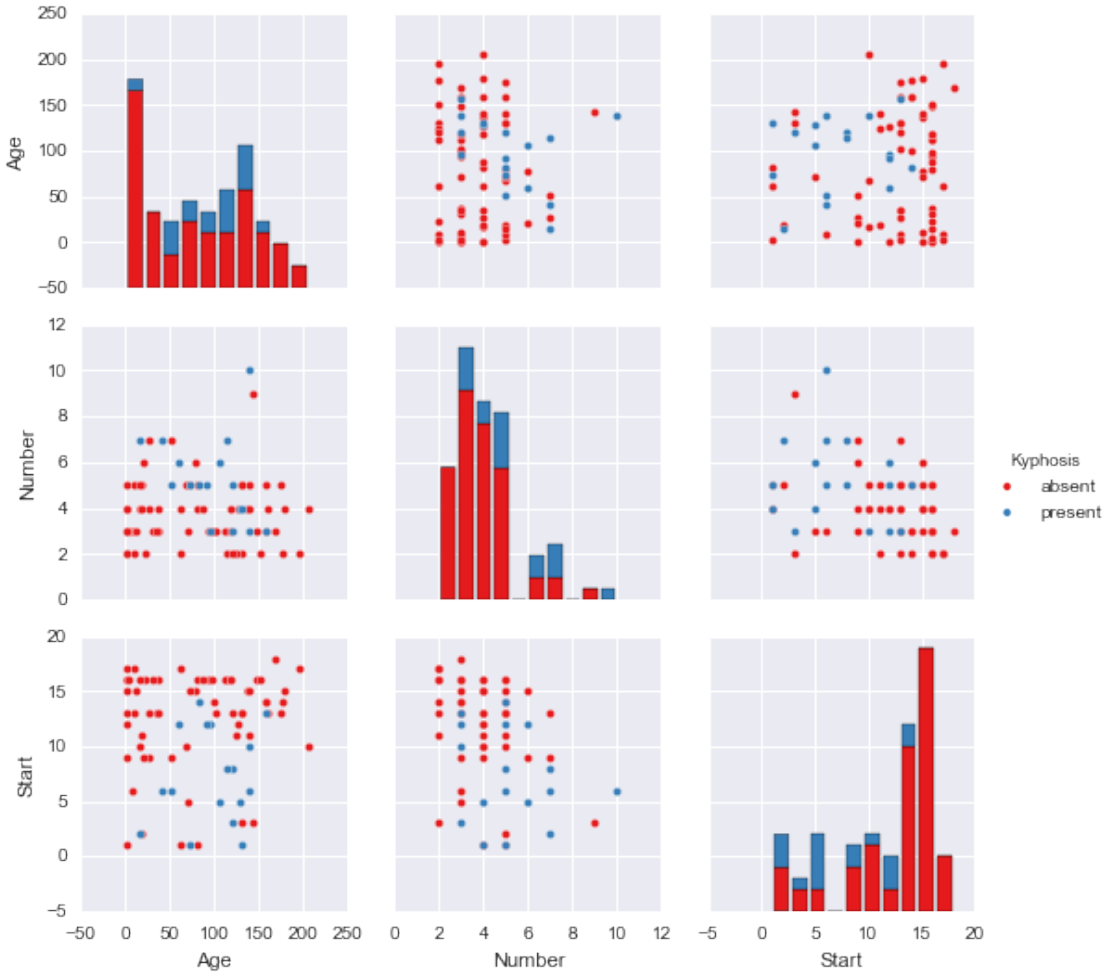
	Kyphosis	Age	Number	Start
0	absent	71	3	5
1	absent	158	3	14
2	present	128	4	5
3	absent	2	5	1
4	absent	1	4	15

1.3 EDA

Vamos observar um simples pairplot para este pequeno conjunto de dados.

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

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



1.4 Divisão treino-teste

Vamos dividir os dados em um conjunto de treinamento e um conjunto de testes!

[4]:
```
from sklearn.model_selection import train_test_split
```

[5]:
```
X = df.drop('Kyphosis',axis=1)
y = df['Kyphosis']
```

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

1.5 Árvores de decisão

Começaremos apenas treinando uma única árvore de decisão.

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

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

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

[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 Previsão e Avaliação

Vamos avaliar a nossa árvore de decisão.

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

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

[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
```

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

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

1.7 Visualização de árvore

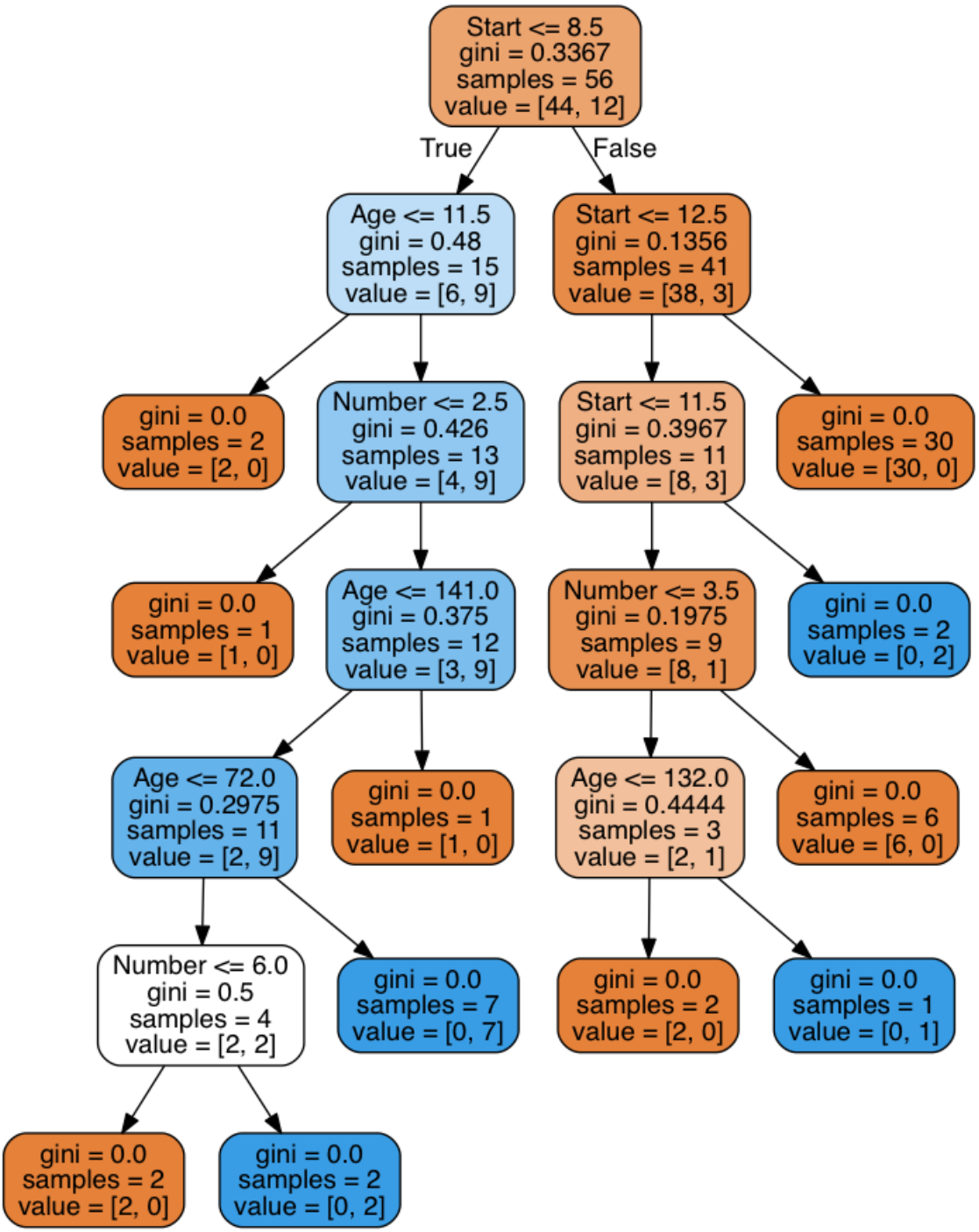
O Scikit learn possui alguns recursos de visualização incorporados para árvores de decisão. Você não usará isso com freqüência e requer que você instale a biblioteca pydot, mas aqui está um exemplo do código para executar isso:

[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
```

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

[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())
```

[39]:



1.8 Florestas aleatórias

Agora vamos comparar o modelo da árvore de decisão com uma floresta aleatória.

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

[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)
```

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

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

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

[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
```