Arvore e Floresta

April 29, 2025

1 Árvores de decisão e florestas aleatórias em Python

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

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

[4]:
```
df.head()
```

[4]:

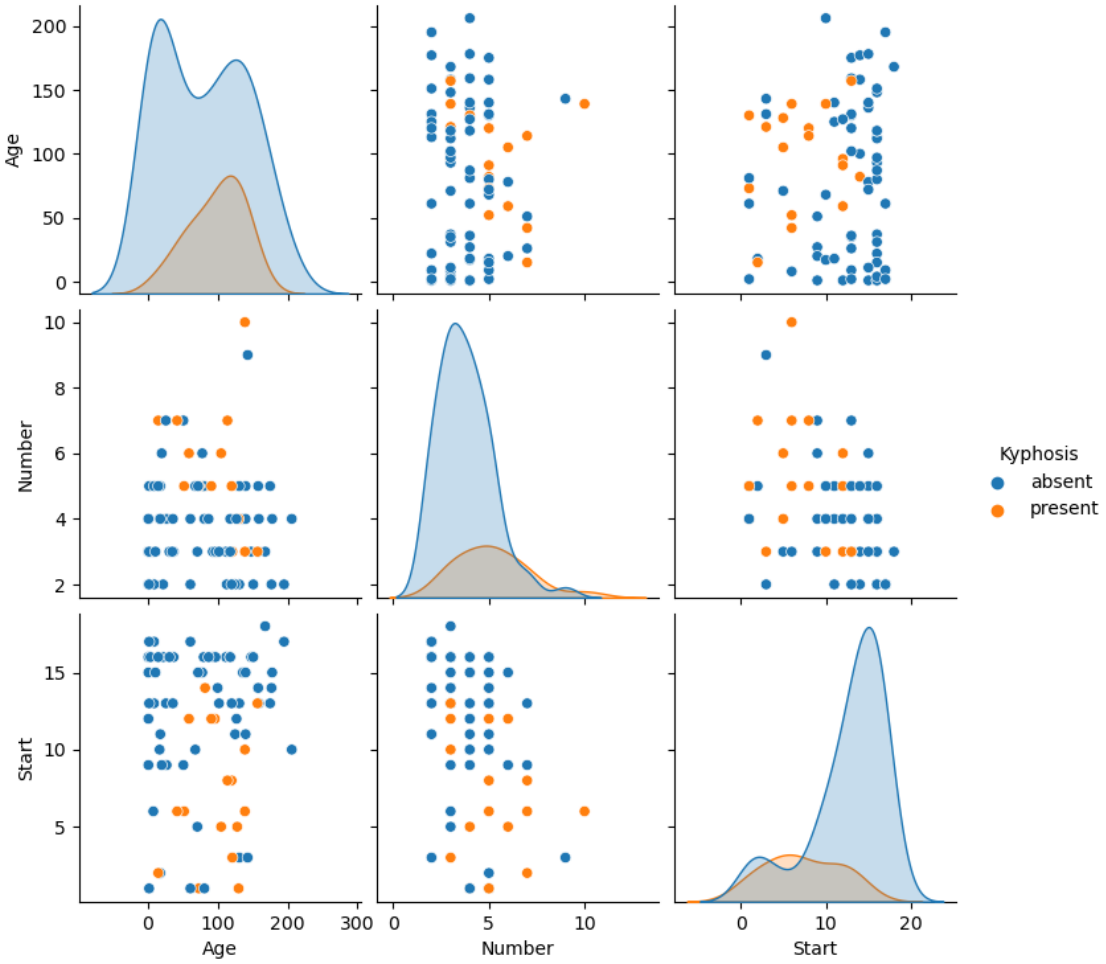
	Kyphosis	Age	Number	Start
0	absent	71	3	5
1	absent	158	3	14
2	present	128	4	5
3	absent	2	5	1
4	absent	1	4	15

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Kyphosis  81 non-null     object
 1   Age       81 non-null     int64
 2   Number    81 non-null     int64
 3   Start     81 non-null     int64
dtypes: int64(3), object(1)
memory usage: 2.7+ KB
```

[6]:
```
sns.pairplot(df, hue='Kyphosis')
```

[6]:
```
<seaborn.axisgrid.PairGrid at 0x7fda7d3f0750>
```



2 Divisão do dataset em treino e teste

[7]:
```
from sklearn.model_selection import train_test_split
```

[8]:
```
df.head()
```

[8]:

	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

[9]:
```
x=df.drop('Kyphosis', axis=1)
```

[10]:
```
y=df['Kyphosis']
```

[11]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.3,
                                                    random_state=42)
```

[21]:
```
X_train.head()
```

[21]:

	Age	Number	Start
62	81	4	1
42	143	9	3
54	140	4	15
16	78	6	15
39	91	5	12

[1]:
```
from sklearn.tree import DecisionTreeClassifier
```

[12]:
```
dtree= DecisionTreeClassifier()
```

[13]:
```
dtree.fit(X_train, y_train)
```

[13]:
```
DecisionTreeClassifier()
```

[14]:
```
predicao=dtree.predict(X_test)
```

[15]:
```
from sklearn.metrics import  classification_report, confusion_matrix
```

[16]:
```
print(classification_report(y_test, predicao))
```

```
              precision    recall  f1-score   support

      absent       0.76      0.84      0.80        19
     present       0.25      0.17      0.20         6

    accuracy                           0.68        25
   macro avg       0.51      0.50      0.50        25
weighted avg       0.64      0.68      0.66        25
```

[17]:
```
print(confusion_matrix(y_test, predicao))
```

```
[[16  3]
 [ 5  1]]
```

[19]:
```
from IPython.display import Image
#from sklearn.externals.six import  StringIO
from io import StringIO
from sklearn.tree import  export_graphviz
import pydot

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

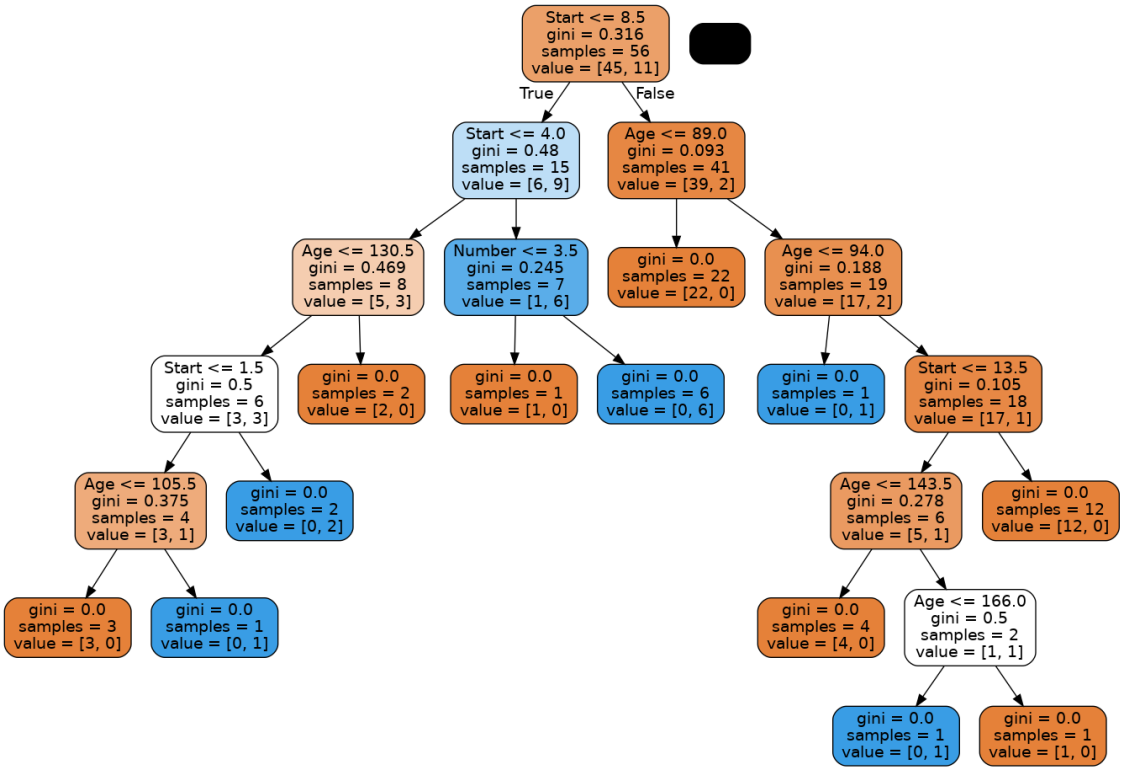
```
features
```

[19]:
```
['Age', 'Number', 'Start']
```

[20]:
```
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())
```

[20]:



3 Floresta Aleatória

[21]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(n_estimators=100)
rfc.fit(X_train,y_train)
```

[21]:
```
RandomForestClassifier()
```

[22]:
```
rfc_pred=rfc.predict(X_test)
```

[23]:
```
print(confusion_matrix(y_test, rfc_pred))
```

```
[[19  0]
 [ 5  1]]
```

[24]:
```
print(classification_report(y_test, rfc_pred))
```

```
              precision    recall  f1-score   support

      absent       0.79      1.00      0.88        19
     present       1.00      0.17      0.29         6

    accuracy                           0.80        25
   macro avg       0.90      0.58      0.58        25
weighted avg       0.84      0.80      0.74        25
```

Bagging model

[49]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score

# Define the model
random_forest_model = RandomForestClassifier()
# Fit the random search object to the data
random_forest_model.fit(X_train, y_train)
```

[49]:
```
RandomForestClassifier()
```

[50]:
```
# Make predictions
y_pred = random_forest_model.predict(X_test)
```

[51]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[19  0]
 [ 5  1]]
```

[31]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

      absent       0.83      1.00      0.91        20
     present       1.00      0.20      0.33         5

    accuracy                           0.84        25
   macro avg       0.92      0.60      0.62        25
weighted avg       0.87      0.84      0.79        25
```

[ ]:
```

```