Decisions Trees and Random Forests with Python

In [2]:
```
#Firstly let's import our libraries
```

In [3]:
```
import pandas as pd
import numpy as np
```

In [5]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:
```
%matplotlib inline
```

In [7]:
```
df = pd.read_csv('kyphosis.csv')
```

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

#The age is the age of the person in month so these are info on children
# Number = number of veterbrates involved in the operation and
# Start = the number of the first or top most veterbrate that was operated on
# Kyphosis is our target
```

Out[8]:

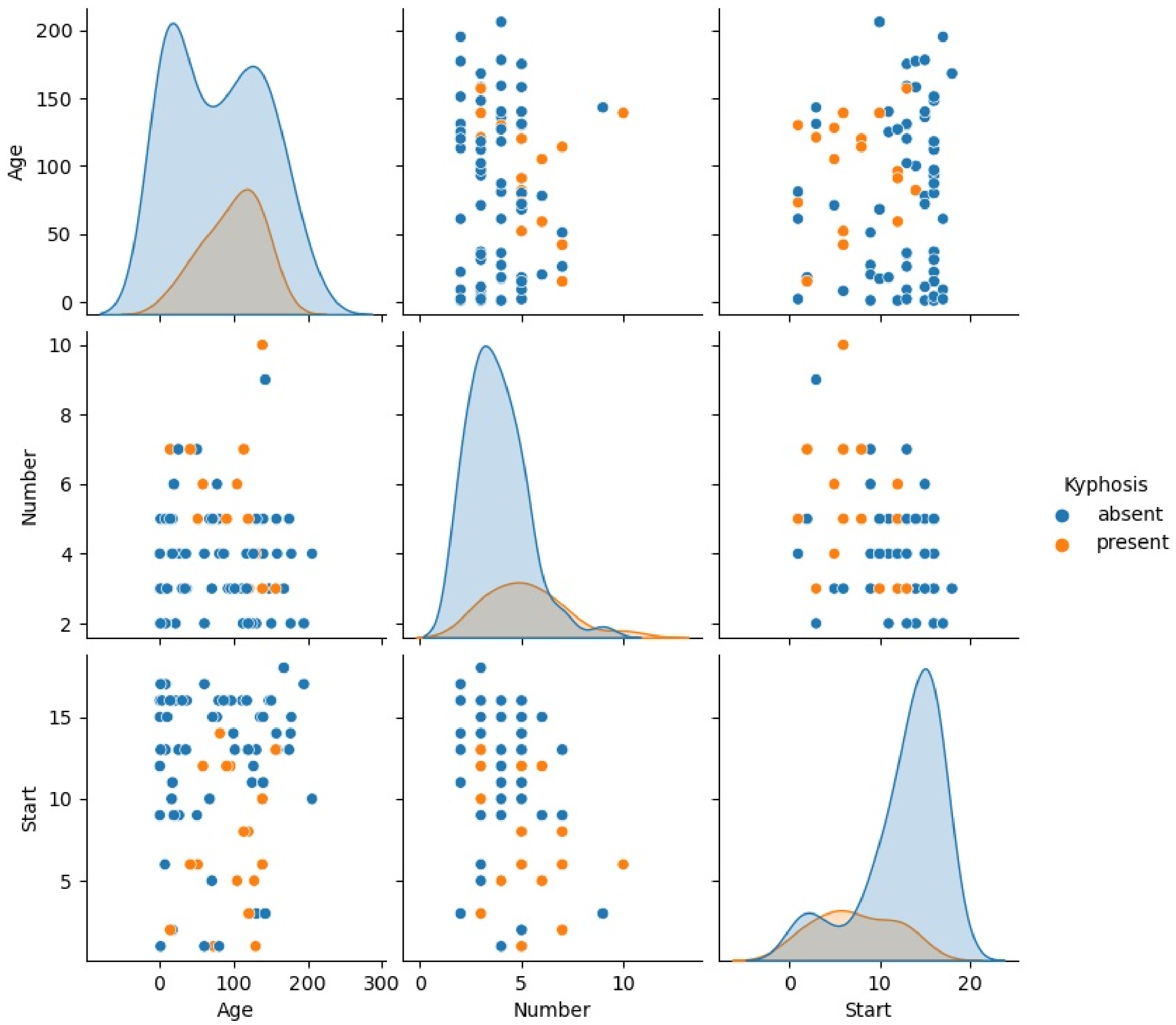
	Kyphosis	Age	Number	Start
0	absent	71	3	5
1	absent	158	3	14
2	present	128	4	5
3	absent	2	5	1
4	absent	1	4	15

In [10]:
```
df.info() #Small data set
```

```
<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
```

In [11]:
```
sns.pairplot(df,hue= 'Kyphosis')
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x15c6983b4f0>
```



In [12]:
```
from sklearn.model_selection import train_test_split #General import - # Step 1
```

In [14]:
```
#Let's set our X data to everything but the target - # Step 2

X = df.drop('Kyphosis',axis=1)

# And the target is our kyphosis column
```

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

In [16]:
```
#Get the whole train test source code. Also let's use default random state - #Step 3

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [18]:
```
#Take our classifier(import what is needed specifally) - #Step 4

from sklearn.tree import DecisionTreeClassifier
```

In [20]:
```
#Define our Decision Tree Classifier - #Step 5

dtree = DecisionTreeClassifier()
```

In [21]:
```
#Fit the model - #Step 6

dtree.fit(X_train,y_train)
```

Out[21]:
```
DecisionTreeClassifier()
```

In [22]:
```
#Set our predictions - #Step 7

predictions = dtree.predict(X_test)
```

In [23]:
```
#Import our classification report and confusion matrix since we didn't do it - #Step 8

from sklearn.metrics import classification_report,confusion_matrix
```

In [25]:
```
#Let's print both of them - #Step 9

print(classification_report(y_test,predictions))
print ('\n')
print(confusion_matrix(y_test,predictions))
```

```
              precision    recall  f1-score   support

      absent       0.83      0.79      0.81        19
     present       0.43      0.50      0.46         6

    accuracy                           0.72        25
   macro avg       0.63      0.64      0.64        25
weighted avg       0.74      0.72      0.73        25



[[15  4]
 [ 3  3]]
```

In [32]:
```
#Now we want to see how these results compare to a random forest model - #Step10

from sklearn.ensemble import RandomForestClassifier
```

In [33]:
```
#Set our Random forest classifier - #Step11

rfc = RandomForestClassifier(n_estimators=200)
```

In [34]:
```
#Let's fit our model to train - #Step12

rfc.fit(X_train,y_train)
```

Out[34]:
```
RandomForestClassifier(n_estimators=200)
```

In [35]:
```
#Set our predictions - #Step13

rfc_pred = rfc.predict(X_test)
```

In [36]:
```
#Print our classification and confusion matrix results out for rfc - #Step14

print(classification_report(y_test,rfc_pred))
print ('\n')
print(confusion_matrix(y_test,rfc_pred))
```

```
              precision    recall  f1-score   support

      absent       0.89      0.89      0.89        19
     present       0.67      0.67      0.67         6

    accuracy                           0.84        25
   macro avg       0.78      0.78      0.78        25
weighted avg       0.84      0.84      0.84        25
```

```
[[17  2]
 [ 2  4]]
```

In [37]:

```
#We can see that the random forest did better than a single decision tree
# check precision and other parameters.
#Most of the time, as datasets get larger the random forest is always going to
#  outperform better and better a single decision tree
```

In [ ]:

```
#Random forest is an extremely powerful tool when it comes to machine learning algorithms
# And a lot of times it's a data scientist first quick choice for recreating
#  a very fast classification model as far as just trying to see what
#   kind of a baseline accuracy or precision or recall you can get a model
#    before you start kind of playing around for other other models or tuning stuff.
```