In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:

```
df=pd.read_csv('kyphosis.csv')
```

In [16]:

```
df.head()
print('\n')
df.info()#this dataset represents whether or not the kyphosis condtion was present or a
bsent after the operation
```

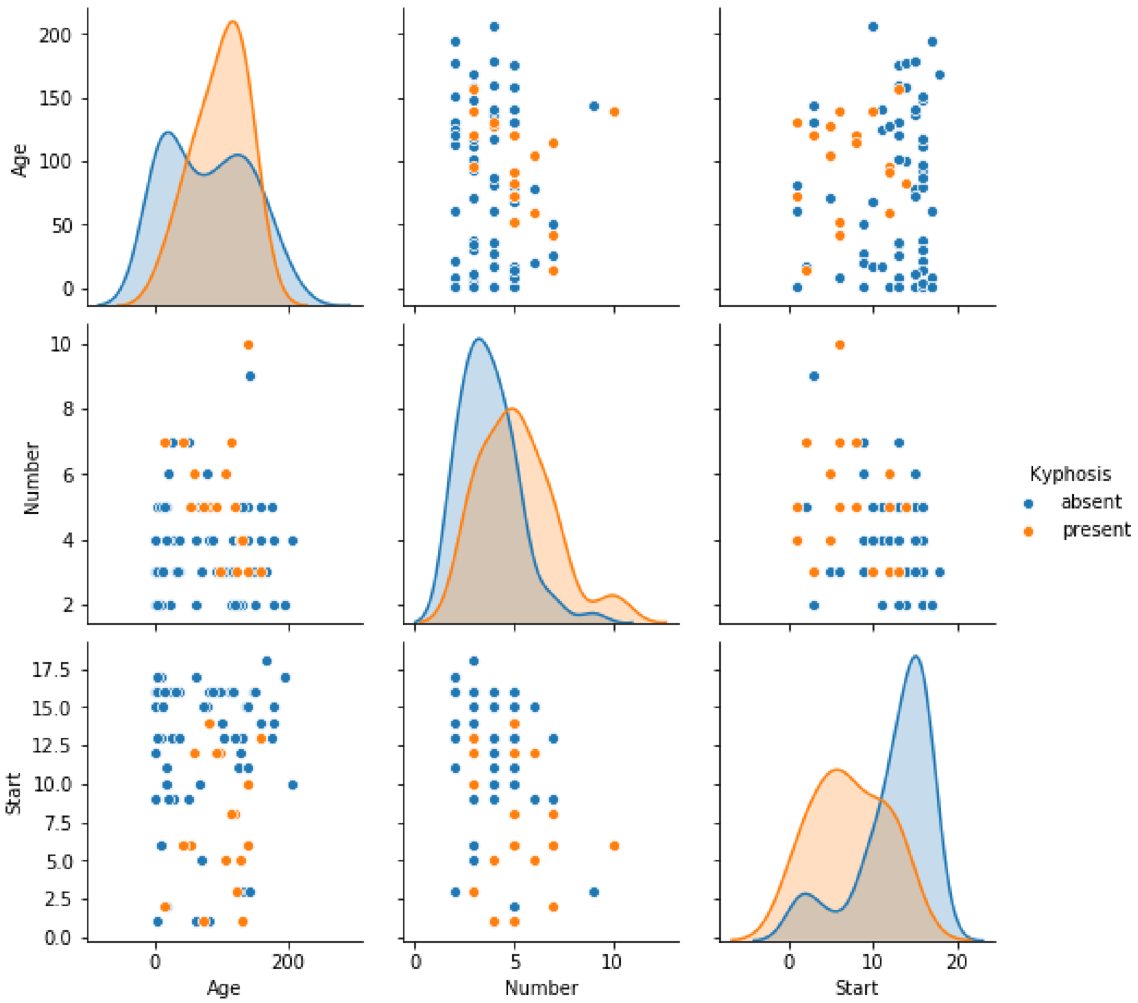
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Kyphosis  81 non-null     object
 1   Age       81 non-null     int64 
 2   Number    81 non-null     int64 
 3  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

In [6]:

```
sns.pairplot(df,hue='Kyphosis')
```

Out[6]:

```
&lt;seaborn.axisgrid.PairGrid at 0x2770f7257c8&gt;
```



In [7]:

```
#moving on to machine learning modelling
from sklearn.model_selection import train_test_split
```

In [8]:

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

In [9]:

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

In [20]:

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

In [12]:

```
#Proceeding to train a single decision tree
```

In [21]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [23]:

```
dtree=DecisionTreeClassifier()
```

In [24]:

```
dtree.fit(X_train,y_train)
```

Out[24]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=N
one,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [25]:

```
predictions=dtree.predict(X_test)
```

In [26]:

```
from sklearn.metrics import classification_report,confusion_matrix
```

In [27]:

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

```
[[18  5]
 [ 1  1]]


              precision    recall  f1-score   support

      absent       0.95      0.78      0.86        23
     present       0.17      0.50      0.25         2

    accuracy                           0.76        25
   macro avg       0.56      0.64      0.55        25
weighted avg       0.88      0.76      0.81        25
```

In [28]:

```
#Lets see how this compares to a random forest model
```

In [29]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [31]:

```
rfc=RandomForestClassifier(n_estimators=200) #probably an overkill
```

In [32]:

```
rfc.fit(X_train,y_train)
```

Out[32]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=200,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [33]:

```
rfc_prediction=rfc.predict(X_test)
```

In [34]:

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

```
[[20  3]
 [ 1  1]]


              precision    recall  f1-score   support

      absent       0.95      0.87      0.91        23
     present       0.25      0.50      0.33         2

    accuracy                           0.84        25
   macro avg       0.60      0.68      0.62        25
weighted avg       0.90      0.84      0.86        25
```

In [35]:

```
#Okay so our accuray have def went up, only 4 FP+FN of 25 data points, we hit a 90%
#WE can see that random forest did better, and we will notice that random forest always
does better when we have larger datasets
```

In [36]:

```
df['Kyphosis'].value_counts()
```

Out[36]:

```
absent     64
present    17
Name: Kyphosis, dtype: int64
```

In [37]:

```
#We can see that the label set itself is kind of unbalanced, so that also affects model
s a lot
```

In [ ]:

```

```