# Decision Trees & Random Forests

## Import Packages

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

%matplotlib inline
```

## Investigate Dataset

In [2]:
```
raw_data = pd.read_csv('kyphosis-data.csv')
```

In [3]:
```
display(raw_data.columns)
```

```
Index(['Kyphosis', 'Age', 'Number', 'Start'], dtype='object')
```

In [4]:
```
display(raw_data)
```

| | 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 |
| **...** | ... | ... | ... | ... |
| **76** | present | 157 | 3 | 13 |
| **77** | absent | 26 | 7 | 13 |
| **78** | absent | 120 | 2 | 13 |
| **79** | present | 42 | 7 | 6 |
| **80** | absent | 36 | 4 | 13 |

81 rows × 4 columns

Look at features included in Dataset.

- `Kyphosis` column contains a value of *present* or *absent* depending on whether the individual had the disease.

- `Age` column contains the patient's age in months.
- `Number` column contains the number of vertebrae involved in operation.
- `Start` column describes top-most vertebrae that was operated on.

## Exploratory Data Analysis

Exploratory Data Analysis usually involves calculating aggregate data or building visualizations.

It's important to understand the size of Dataset for Machine Learning Engineers.
The `pandas` library method `info()` can be invoked on a DataFrame to let you know the number of observations in the Dataset. (e.g. It should be *81* for our relatively small Dataset.)

In [5]:
```
raw_data.info()
```

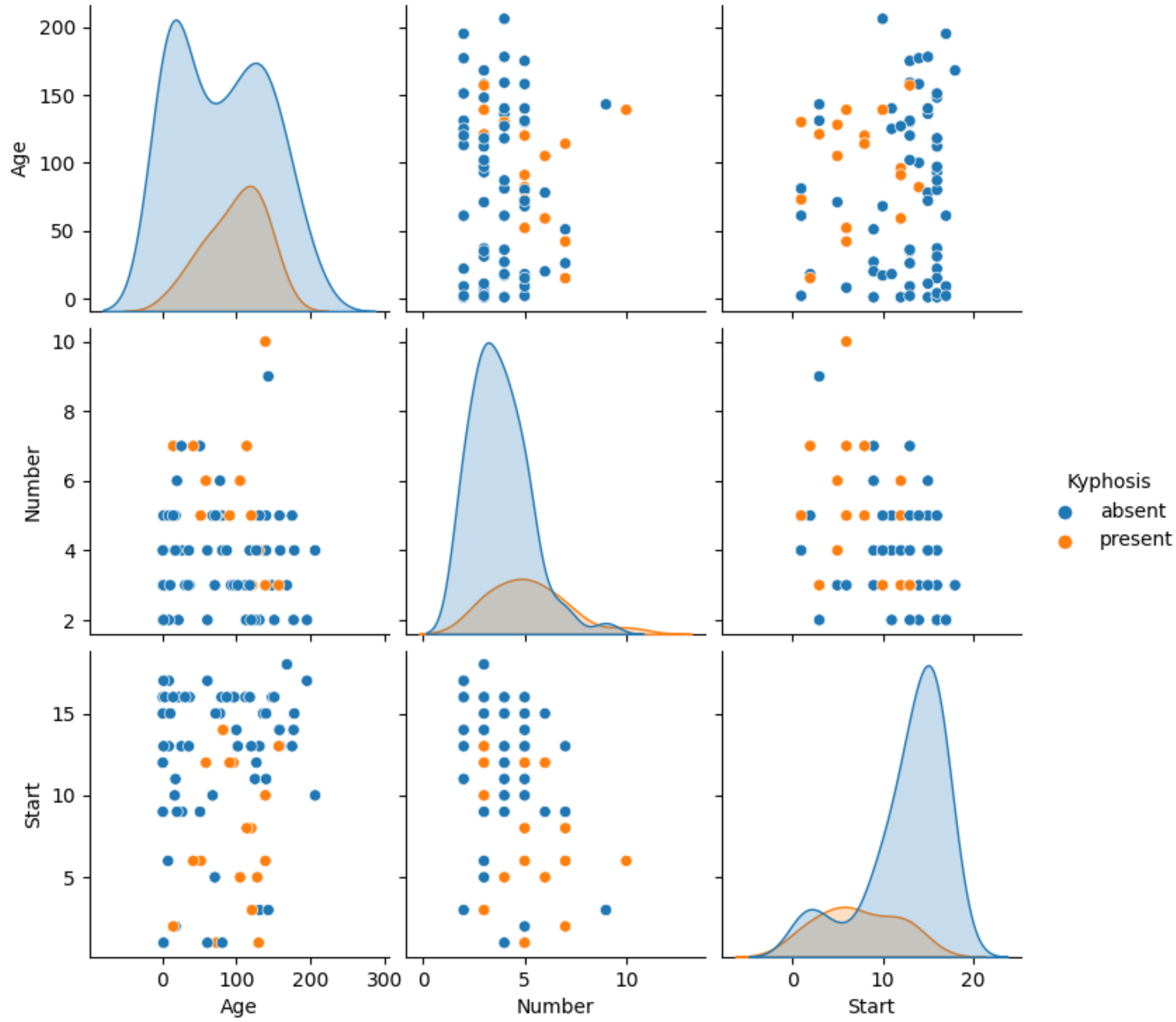
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Kyphosis  81 non-null     object
 1   Age       81 non-null     int64 
 2   Number    81 non-null     int64 
 3  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

### Visualize Dataset

Use `seaborn` library to generate a pairplot and visualize what's happening with each feature.

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

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x1aee395f130>
```




## Split Training and Test Data

Use a test size of 30% to train our model.

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

In [8]:
```
# Specify x-values and y-values
x = raw_data.drop('Kyphosis', axis = 1)
y = raw_data['Kyphosis']
```

In [9]:
```
x_training_data, x_test_data, y_training_data, y_test_data = train_test_split(x, y,
```

# Decision Tree Model

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

In [11]:
```
decision_tree_model = DecisionTreeClassifier().fit(x_training_data, y_training_data
decision_tree_predictions = decision_tree_model.predict(x_test_data)
```

## Evaluate Model Performance

In [12]:
```python
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

In [13]:
```python
decision_tree_report = classification_report(y_test_data, decision_tree_predictions
print(decision_tree_report)
```

```
              precision    recall  f1-score   support

      absent       0.77      0.94      0.85        18
     present       0.67      0.29      0.40         7

    accuracy                           0.76        25
   macro avg       0.72      0.62      0.62        25
weighted avg       0.74      0.76      0.72        25
```

In [19]:
```python
decision_tree_matrix = confusion_matrix(y_test_data, decision_tree_predictions)
print(decision_tree_matrix)
```

```
[[17  1]
 [ 5  2]]
```

We can see that our model has incorect predictions on 5 data points :

- *2* False Positives
- *3* False Negatives

# Random Forest Model

In [15]:
```python
from sklearn.ensemble import RandomForestClassifier
```

In [16]:
```python
random_forest_model = RandomForestClassifier().fit(x_training_data, y_training_data
random_forest_predictions = random_forest_model.predict(x_test_data)
```

In [17]:
```python
random_forest_report = classification_report(y_test_data, random_forest_predictions
print(random_forest_report)
```

```
              precision    recall  f1-score   support

      absent       0.75      1.00      0.86        18
     present       1.00      0.14      0.25         7

    accuracy                           0.76        25
   macro avg       0.88      0.57      0.55        25
weighted avg       0.82      0.76      0.69        25
```

In [18]:
```python
random_forest_matrix = confusion_matrix(y_test_data, random_forest_predictions)
print(random_forest_matrix)
```

```
array([[18,  0],
       [ 6,  1]], dtype=int64)
```

The **Random Forest** Model hasn't performed significantly better than the **Decision Tree** Model. This is due to the small size of our Dataset.

*It is extremely likely for **Random Forests** to perform better than basic **Decision Trees** on larger Datasets.*