Kyphosis Disease Classification Project

IMPORT LIBRARIES AND DATASETS

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
import pickle
```

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

Out[2]:

	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

In [3]:
```
kyphosis_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
```

In [4]:
```
kyphosis_df.describe()
```

Out[4]:

	Age	Number	Start
count	81.000000	81.000000	81.000000
mean	83.654321	4.049383	11.493827
std	58.104251	1.619423	4.883962
min	1.000000	2.000000	1.000000
25%	26.000000	3.000000	9.000000
50%	87.000000	4.000000	13.000000
75%	130.000000	5.000000	16.000000
max	206.000000	10.000000	18.000000

PERFORM DATA VISUALIZATION

In [5]:
```
LabelEncoder_y = LabelEncoder()
kyphosis_df['Kyphosis'] = LabelEncoder_y.fit_transform(kyphosis_df['Kyphosis'])
```

In [6]:
```
kyphosis_df
```

Out[6]:

	Kyphosis	Age	Number	Start
0	0	71	3	5
1	0	158	3	14
2	1	128	4	5
3	0	2	5	1
4	0	1	4	15
...	...	...	...	...
76	1	157	3	13
77	0	26	7	13
78	0	120	2	13
79	1	42	7	6
80	0	36	4	13

81 rows × 4 columns

In [7]:
```
kyphosis_True = kyphosis_df[kyphosis_df['Kyphosis']==1]
```

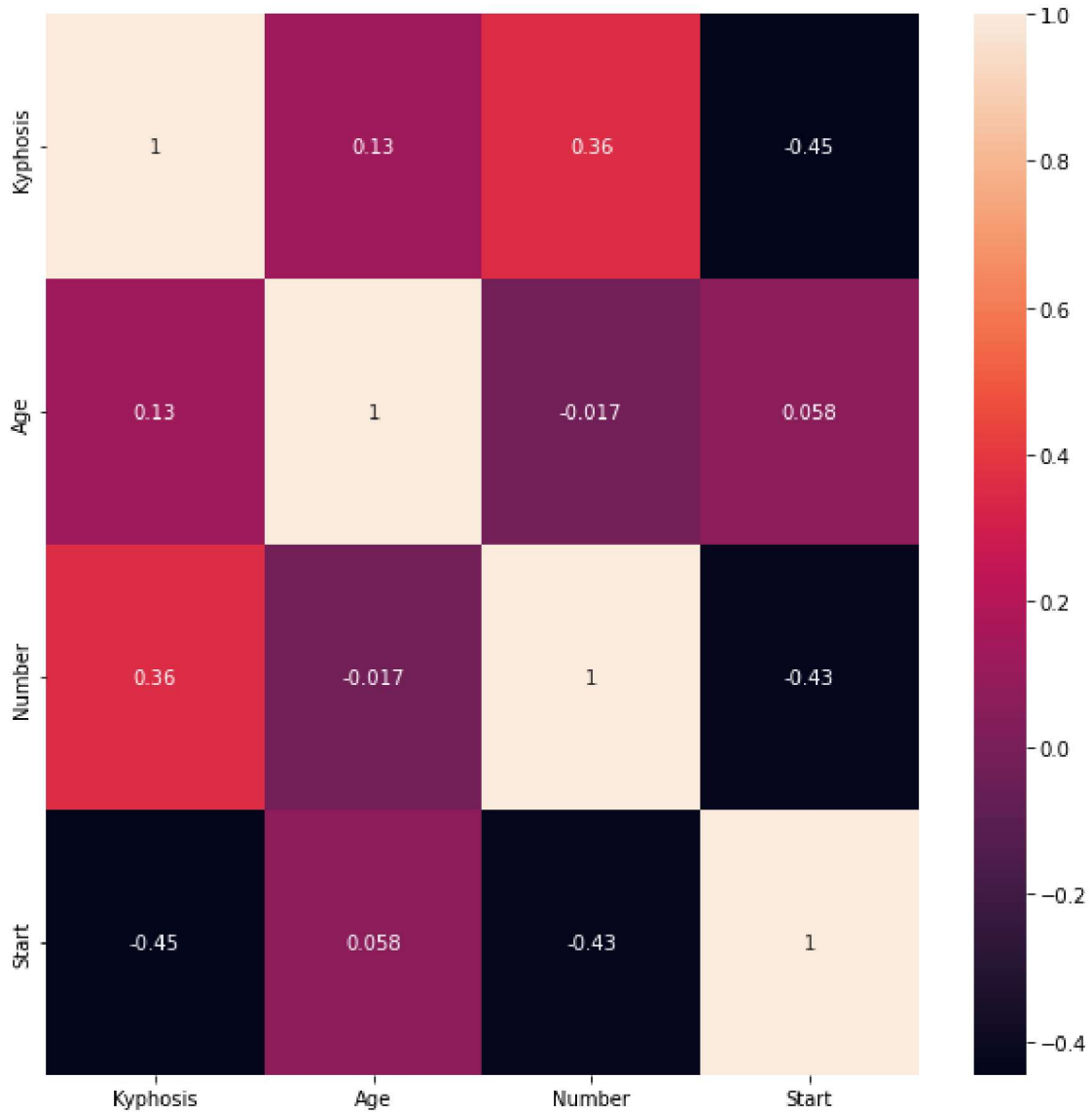
In [8]:
```
kyphosis_False = kyphosis_df[kyphosis_df['Kyphosis']==0]
```

In [9]:
```
print( 'Disease present after operation percentage =', (len(kyphosis_True) / len(
```

```
Disease present after operation percentage = 20.98765432098765 %
```

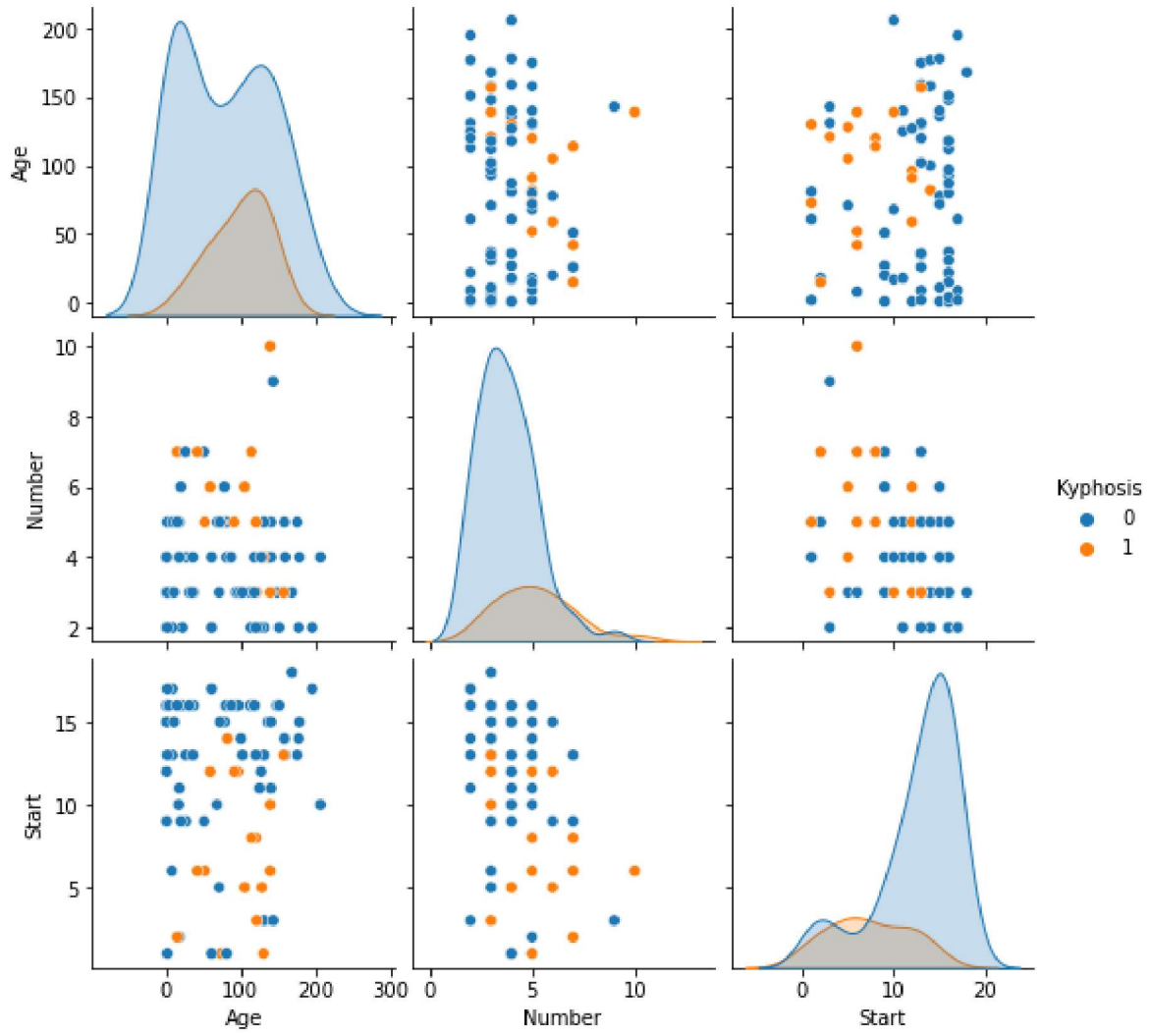
In [10]:
```
plt.figure(figsize=(10,10))
sns.heatmap(kyphosis_df.corr(), annot= True)
```

Out[10]: <AxesSubplot:>



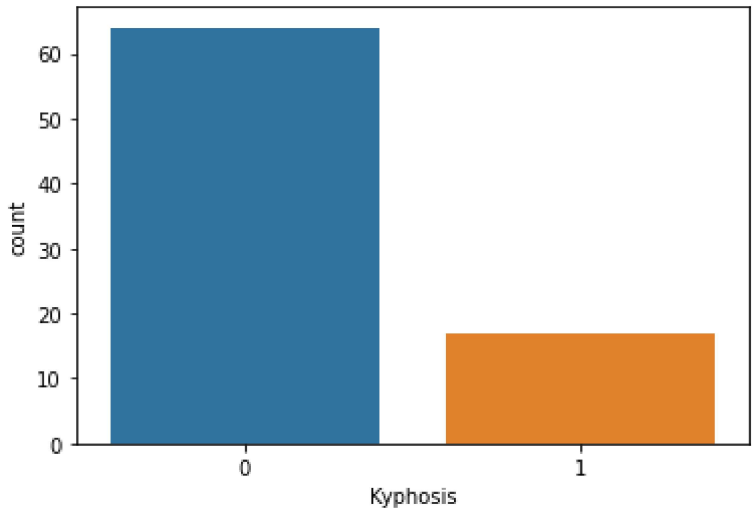
In [11]:
```
sns.pairplot(kyphosis_df, hue= 'Kyphosis')
```

Out[11]: <seaborn.axisgrid.PairGrid at 0x297cbe7a730>



Plotting the data countplot showing how many samples belong to each class

In [12]:
```
sns.countplot(x = kyphosis_df['Kyphosis'], label = "Count");
```



TASK #4: CREATE TESTING AND TRAINING DATASET/DATA CLEANING

In [13]:
```
# Let's drop the target label coloumns
X = kyphosis_df.drop(['Kyphosis'], axis = 1)
Y = kyphosis_df['Kyphosis']
```

In [14]:
```
X
```

Out[14]:

	Age	Number	Start
0	71	3	5
1	158	3	14
2	128	4	5
3	2	5	1
4	1	4	15
...	...	...	...
76	157	3	13
77	26	7	13
78	120	2	13
79	42	7	6
80	36	4	13

81 rows × 3 columns

In [15]:
```
Y
```

Out[15]:
```
0     0
1     0
2     1
3     0
4     0
     ..
76    1
77    0
78    0
79    1
80    0
Name: Kyphosis, Length: 81, dtype: int32
```

In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size = 0.2)
```

In [18]:
```
X_train.shape
```

Out[18]:
```
(64, 3)
```

In [19]:
```
X_test.shape
```

Out[19]:
```
(17, 3)
```

In [20]:
```
# from sklearn.preprocessing import StandardScaler
# sc = StandardScaler()
# X_train = sc.fit_transform(X_train)
# X_test = sc.transform(X_test)
```

TASK #5: TRAIN A LOGISTIC REGRESSION CLASSIFIER MODEL

In [21]:
```
X_train.shape
```

Out[21]:
```
(64, 3)
```

In [22]:
```
Y_train.shape
```

Out[22]:
```
(64,)
```

In [23]:
```
X_test.shape
```

Out[23]:
```
(17, 3)
```

In [24]:
```
Y_test.shape
```

Out[24]:
```
(17,)
```

In [25]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, Y_train)
```

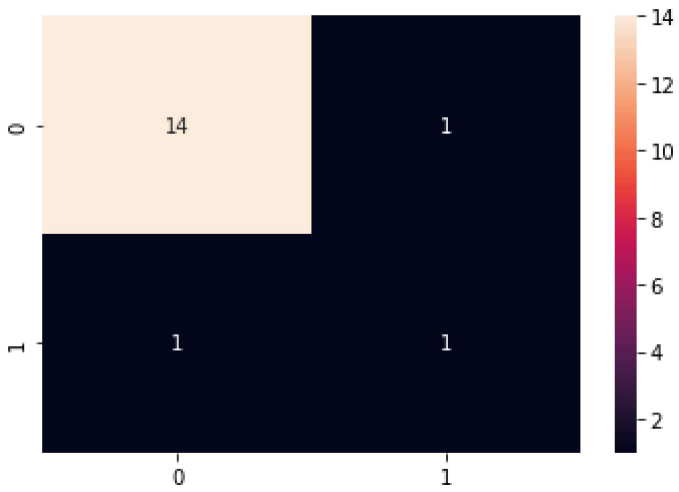
Out[25]:
```
LogisticRegression()
```

TASK #6: EVALUATE TRAINED MODEL PERFORMANCE

In [26]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [27]:
```
# Predicting the Test set results
y_predict_test = model.predict(X_test)
cm = confusion_matrix(Y_test, y_predict_test)
sns.heatmap(cm, annot = True)
```

Out[27]: <AxesSubplot:>



In [28]:
```
print(classification_report(Y_test, y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        15
           1       0.50      0.50      0.50         2

    accuracy                           0.88        17
   macro avg       0.72      0.72      0.72        17
weighted avg       0.88      0.88      0.88        17
```

#IMPLEMENTING DECISION TREES AND RANDOM FOREST CLASSIFIER MODELS

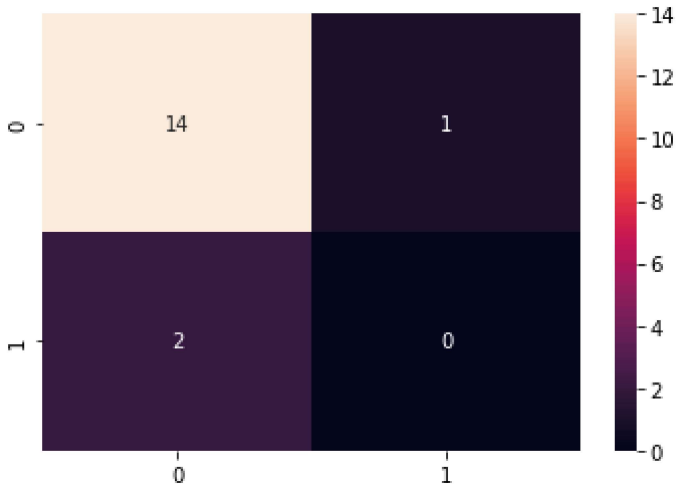
#IMPROVING THE MODEL

In [29]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
```

Out[29]:
```
DecisionTreeClassifier()
```

In [30]:
```
# Predicting the Test set results
y_predict_test = decision_tree.predict(X_test)
cm = confusion_matrix(Y_test, y_predict_test)
sns.heatmap(cm, annot=True)
```

Out[30]:
```
<AxesSubplot:>
```



In [31]:
```
print(classification_report(Y_test, y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.88      0.93      0.90        15
           1       0.00      0.00      0.00         2

    accuracy                           0.82        17
   macro avg       0.44      0.47      0.45        17
weighted avg       0.77      0.82      0.80        17
```

In [32]:
```
feature_importances = pd.DataFrame(decision_tree.feature_importances_,
                                   index = X_train.columns,
                                    columns=['importance']).sort_values('importar

print(feature_importances)
```

```
        importance
Age       0.600095
Start     0.359270
Number    0.040635
```

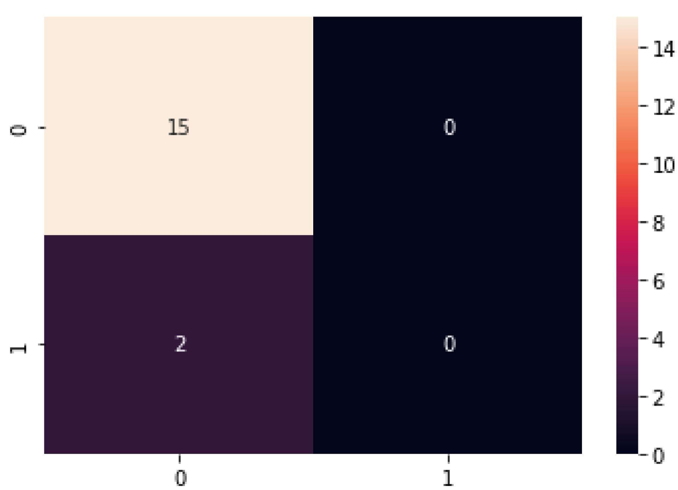
#Implementing the Random Forest Classifier

In [33]:
```
from sklearn.ensemble import RandomForestClassifier
RandomForest = RandomForestClassifier()
RandomForest.fit(X_train, Y_train)
```

Out[33]:
```
RandomForestClassifier()
```

In [34]:
```
# Predicting the Test set results
y_predict_test = RandomForest.predict(X_test)
cm = confusion_matrix(Y_test, y_predict_test)
sns.heatmap(cm, annot=True)

print(classification_report(Y_test, y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        15
           1       0.00      0.00      0.00         2

    accuracy                           0.88        17
   macro avg       0.44      0.50      0.47        17
weighted avg       0.78      0.88      0.83        17


C:\Users\DARSH KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\DARSH KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\DARSH KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



In [36]:
```
#creating a pickel file
pickle.dump(model, open('randomtree.pkl', 'wb'))
```