DECISION TREE :

PAGE 14

NOTE :

PAGE 18

In [1]:
```
# DECISION TREE :
# A) Decision Tree is of Two Types : (1) Classification Tree and (2) Regression Tree.
# B) This particular Classification (or) Regression Selected based on 'Output'.
# C) Decision Tree is a Type of Supervised Learning approach. Mostly used in Classification Problems,
#    but it is used in Regression as well.
# D) Decision Tree Works fine for Both Categorical Variable and Continuous input as Well.
# E) Decision Tree is very Similar to 'Search Trees'.
# F) Decision Tree 'Split the Data/Population' into '2' or more Homogenious Sets(Root node, Decision node, Leaf node)
```

RANDOM FOREST :

PAGE 18

In [2]:
```
# Combination of Multiple Decision Tree Call as 'Random Forest'.
```

A) Bagging: Boot Strup Aggregation :

page 19

In [3]:
```
# One problem with Bagging : The Constructed trees are highly correlated.

# -> Why do Correlation Occur ?
# Because, Every dataset has a Strong Predictor/(or) Feature.
# All the Bagged Trees tend to make the Same Features!!!.
# Because of this all of these Trees look very Similar.

# Correlated Trees - Because, We use all the Features.
```

B) Random Forest Classifier :

In [4]:
```
# Better than 'Bagging' : This Algorithm 'De-Correlates' the Single Decision Trees, that has been Constructed.
# This Reduces the Variance even more When 'Averaging the Trees'.
# Similar to Bagging : We keep Construction Decision Trees on the Training Data, But on Every Split on this Tree,
# a Random Selection of Features/Predictors is Chosen from the Full Feature Set.
# The No.of Features Considered at a given 'Split' is approximately equal to the Square Root
# of the total number of Features(for Classification).
```

Bagging :

page 20

In [5]:
```
# Algorithm searches over all the 'N' Features to find the Best Feature, that Best splits the data at that node.
```

Random Forest Classifier :

In [6]:
```
# Algorith Searches over a Random 'Root N' Features to find the best one.
```

In [7]:
```
# C) RANDOM FOREST CLASSIFIER DIAGRAM : in my Notebook :
```

Now Required Libraries :

page 21

In [8]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [9]:
```
# Now there should be a Data Called 'Kphosis(it's an a Disease)'
# In Kyphosis - Absent,present and also data we have - Age, Number, Start, Something we have the data.
# Now let's call 'Kyphosis' Data
```

In [10]:
```
df = pd.read_csv('DataS/kyphosis.csv')
```

In [11]:
```
# Now the Data has been Loaded Successfully :

df.head()
```

Out[11]:

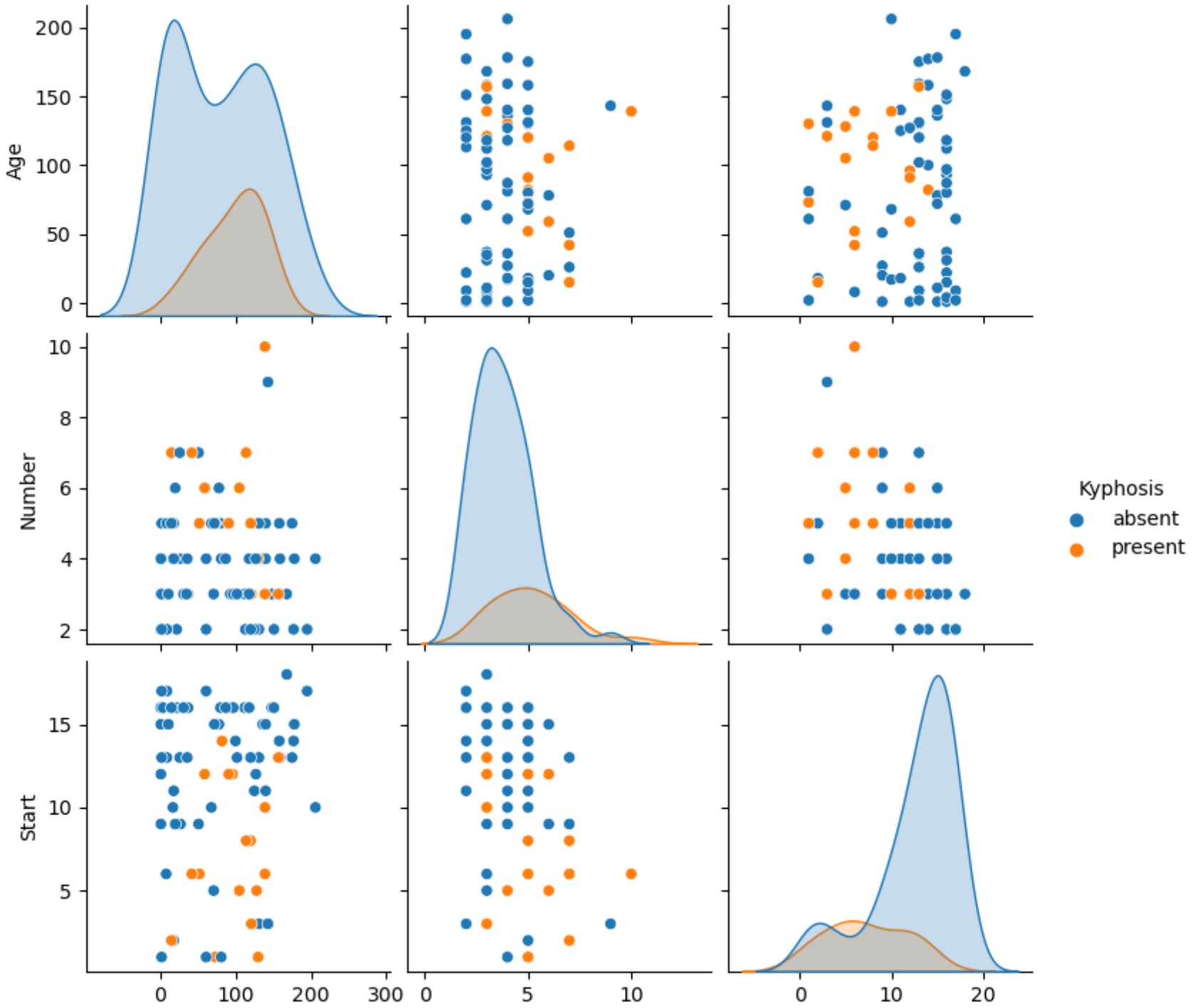
	Kyphosis	Age	Number	Start
0	absent	71	3	5
1	absent	158	3	14
2	present	128	4	5
3	absent	2	5	1
4	absent	1	4	15

In [12]:
```
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 [13]:
```
# Now 'pairplot' of an a particular 'df', and type of Data, hue = 'kyphosis' applied :
# 'K' Capital in the word 'Kyphosis' :
# Now the data, it looks like below image :

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

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x18334ce1fa0>
```


Age
Number
Start
Kyphosis
absent
present
200
150
100
50
0
10
8
6
4
2
15
10
5
0
100
200
300
0
5
10
0
10
20

Age Number Start

In [14]:
```
# Now we need to Call Train_Test_Split :
```

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
# Now i'm Defining 'X' and 'y' :

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

In [17]:
```
# Now, i'm Splittin Capital 'X' here and Let's Call train_test_split :
# Now i'm Declaring Capital 'X' and small 'y' up to this Data :
# NOw the Data Declared :

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [18]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [19]:
```
# Now Here We are taking DecisionTreeClassifier :

dtree = DecisionTreeClassifier()
```

In [20]:
```
# Now Here dtree.modelfit
# Means we are training the Machine :

dtree.fit(X_train, y_train)
```

Out[20]:
```
DecisionTreeClassifier()
```

In [21]:
```
# Now we are Predicting the Data - 'test_data' :

prediction = dtree.predict(X_test)
```

In [22]:
```
# Once we take an a Prediction, Now 'ensemble' RandomForestClassifier.
# In RandomForest How many Trees We Required -> We need to Mention :


from sklearn.ensemble import RandomForestClassifier
```

In [23]:
```
rfc = RandomForestClassifier(n_estimators = 200)
```

In [24]:
```
rfc.fit(X_train, y_train)
```

Out[24]:
```
RandomForestClassifier(n_estimators=200)
```

In [25]:
```
rfc_pred = rfc.predict(X_test)
```

In [26]:
```
# Now i need to get Matrix :

from sklearn.metrics import classification_report, confusion_matrix
```

In [27]:
```
# Now i can go for the predict print both Confusion matrix and Classification report :

print(confusion_matrix(y_test, rfc_pred))
print(classification_report(y_test, rfc_pred))
```

```
[[20  1]
 [ 5  1]]
              precision    recall  f1-score   support

      absent       0.80      0.95      0.87        21
     present       0.50      0.17      0.25         6

    accuracy                           0.78        27
   macro avg       0.65      0.56      0.56        27
weighted avg       0.73      0.78      0.73        27
```

GRAPHICLE REPRESENTATION : INSTALLING ipython Graphicz :

page 25

(A) Now i want to work with an a Graphical Representation :

So, we need to install :

1) pip install ipython,

2) conda install -c anaconda graphviz(Y/N),

3) pip install pydot

in Anaconda powershell prompt(anaconda 3)

(B) Now i required libraries, So i'm calling particular libraries :

In [28]:
```
from IPython.display import Image

# from sklearn.externals.six import StringIO (This function not working)

from sklearn.tree import export_graphviz
import pydot
```

(C) Now i'm taking 'Features' :

page 26

In [29]:
```
features = list (df.columns[1:])  # 'df' data is there already.
features
```

Out[29]:
```
['Age', 'Number', 'Start']
```

In [30]:
```
# Wheather the 'StringIO' is called (or) not i'm not sure, Because We didn't declare StringIO in 33rc Sum here :
```

In [31]:
```
# It not worked due to 'String' not defined :

dot_data = StringIO()     # StringIO - 'S' Capital :
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13404\1897005644.py in <module>
      1 # It not worked due to 'String' not defined :
      2
----> 3 dot_data = StringIO()     # StringIO - 'S' Capital :

NameError: name 'StringIO' is not defined
```

In [ ]:
```
# Then How to define -> StringIO means,
# Instead of StringIO, Let's try 'directly',
# from 'export' - We have called the 'Tree' in 33rd Sum here :
```

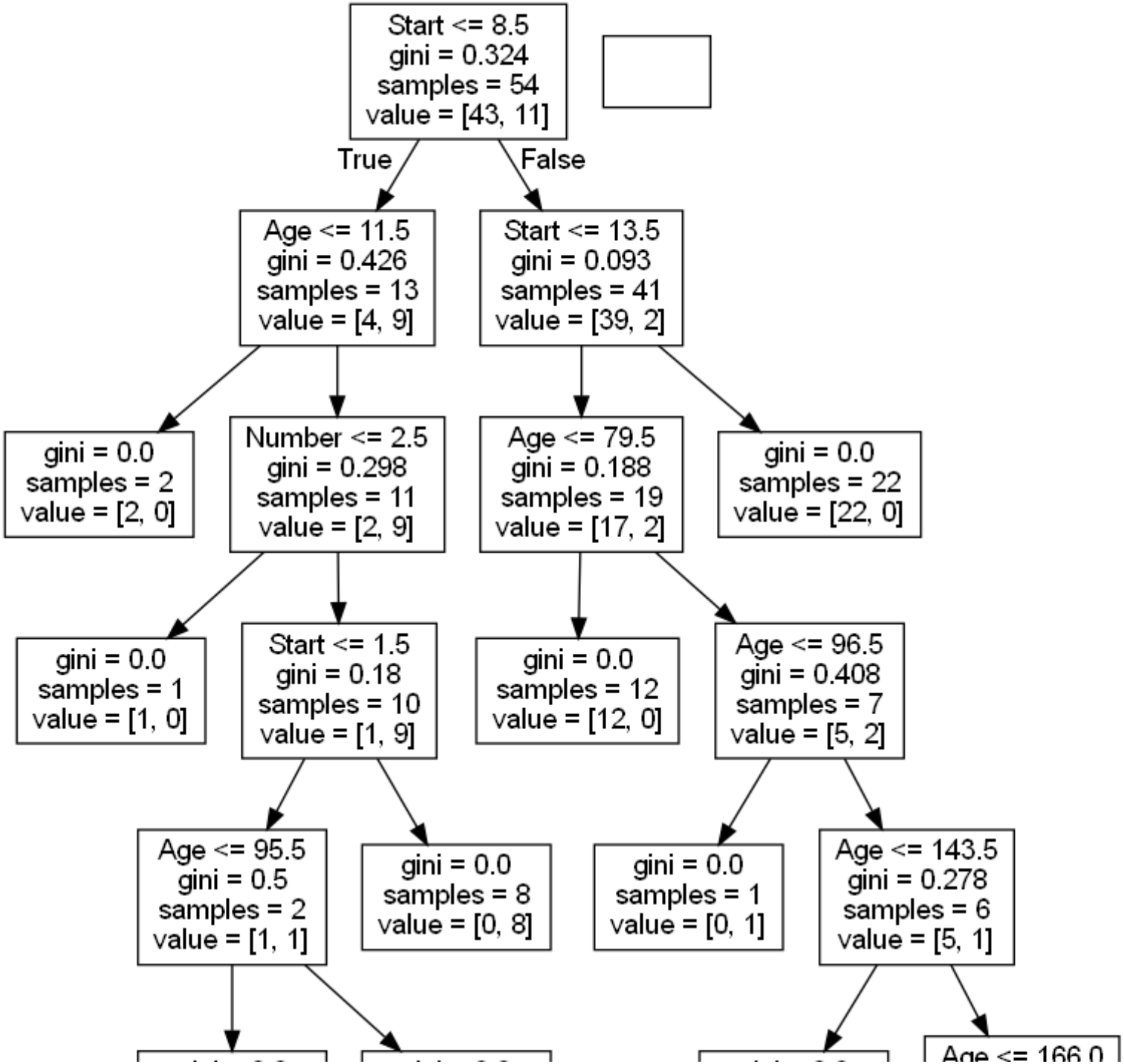
In [33]:
```
from sklearn import tree
```

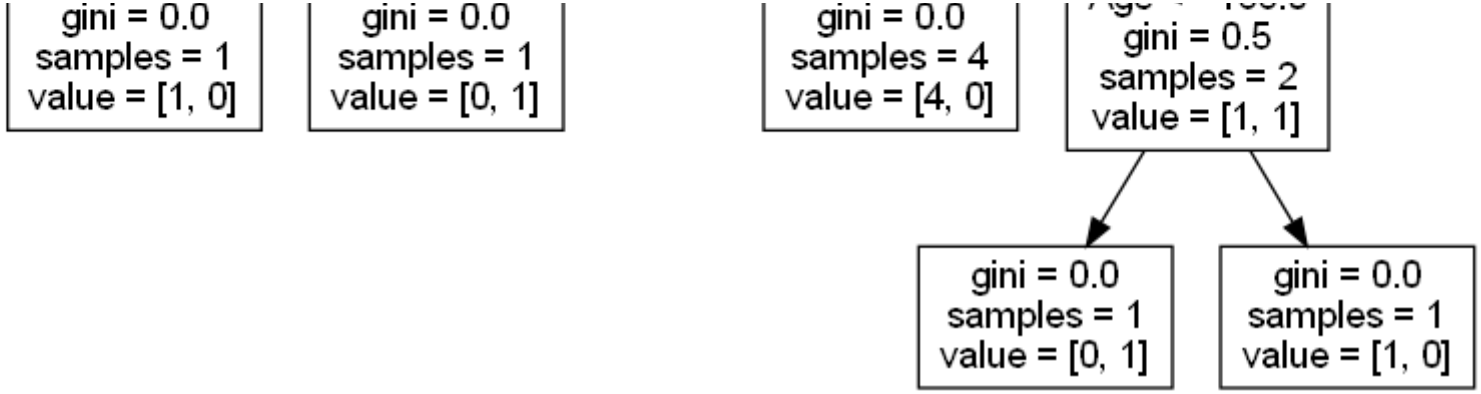
In [34]:
```
# Here in 'None' - 'N' always Capital,
# and 'feature_names' - This is a 'One of the Parameter' :

dot_data = tree.export_graphviz(dtree, out_file = None, feature_names = features)
```

In [35]:
```
# Here, What exactly means, We called 'pydot' already in 33rd Sum. So, No issue :
```

In [37]:
```
# Here, 'graph[0] - Base '0' Position' :
# Based on our Samples, it is going to be taken our Data :

graph = pydot.graph_from_dot_data(dot_data)
Image(graph[0].create_png())
```

Out[37]:


Start <= 8.5
gini = 0.324
samples = 54
value = [43, 11]
True
False
Age <= 11.5
gini = 0.426
samples = 13
value = [4, 9]
Start <= 13.5
gini = 0.093
samples = 41
value = [39, 2]
gini = 0.0
samples = 2
value = [2, 0]
Number <= 2.5
gini = 0.298
samples = 11
value = [2, 9]
Age <= 79.5
gini = 0.188
samples = 19
value = [17, 2]
gini = 0.0
samples = 22
value = [22, 0]
gini = 0.0
samples = 1
value = [1, 0]
Start <= 1.5
gini = 0.18
samples = 10
value = [1, 9]
gini = 0.0
samples = 12
value = [12, 0]
Age <= 96.5
gini = 0.408
samples = 7
value = [5, 2]
Age <= 95.5
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 8
value = [0, 8]
gini = 0.0
samples = 1
value = [0, 1]
Age <= 143.5
gini = 0.278
samples = 6
value = [5, 1]


gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 4
value = [4, 0]
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 1
value = [1, 0]

In [ ]: