Machine Learning: Decision Tree 2

Serigne Fallou MBacke NGOM

2023-11-22

Reading Data

```
import numpy as np
import pandas as pd

data = pd.read_csv('cancerdata1.csv')
data.loc[np.r_[0:3, 51:53, 101:103], :]
```

```
##      ESR1  PGR  BCL2  NAT1     Results
## 0     5.1  3.5   1.4   0.2       Cured
## 1     4.9  3.0   1.4   0.2       Cured
## 2     4.7  3.2   1.3   0.2       Cured
## 51    6.4  3.2   4.5   1.5  Recurrence
## 52    6.9  3.1   4.9   1.5  Recurrence
## 101   5.8  2.7   5.1   1.9        Dead
## 102   7.1  3.0   5.9   2.1        Dead
```

Data Check (Missing Values, Label Check)

- Check the dataset to make sure no data is missing and Check the class labels;
- Use data_found as a dummy variable to determine whether to print missing value information

```
def verify_dataset(data):
    data_found = 1
    for each_column in data.columns:
        if data[each_column].isnull().any():
            print("Data missing in Column " + each_column)
            data_found = 0
            quit()

        if data_found == 1:
            print("Dataset is complete. No missing value")

        return

verify_dataset(data)
```

```
## Dataset is complete. No missing value
```

Create a Training & Testing Set

The data set of 150 inds is divided into 70% for training and 30% for testing (3-fold cross validation): - Splitting The Datase in training and testing; - Use the .sample() function to scramble the data set; - Determine the integer location (iloc) from beginning of array (:) to 0.7*150 and do a ”cleanup” with a reset call; - Call split_dataset_test_train and check data sets.

```
def split_dataset_test_train(data):
    data = data.sample(frac=1).reset_index(drop=True)
    training_data = data.iloc[:int(0.7 * len(data))].reset_index(drop=True)
    testing_data = data.iloc[int(0.7 * len(data)):].reset_index(drop=True)
    return [training_data, testing_data]

testtrain = split_dataset_test_train(data)
print(testtrain)
```

```
## [     ESR1  PGR  BCL2  NAT1      Results
## 0     5.8  2.6   4.0   1.2  Recurrence
## 1     5.0  2.0   3.5   1.0  Recurrence
## 2     5.4  3.4   1.5   0.4       Cured
## 3     6.7  3.1   4.7   1.5  Recurrence
## 4     5.5  2.6   4.4   1.2  Recurrence
## ..    ...  ...   ...   ...         ...
## 100   4.9  3.1   1.5   0.1       Cured
## 101   6.1  2.8   4.0   1.3  Recurrence
## 102   4.8  3.0   1.4   0.1       Cured
## 103   5.2  2.7   3.9   1.4  Recurrence
## 104   6.5  3.0   5.8   2.2        Dead
##
## [105 rows x 5 columns],     ESR1  PGR  BCL2  NAT1      Results
## 0    6.0  3.4   4.5   1.6  Recurrence
## 1    5.8  2.8   5.1   2.4        Dead
## 2    5.1  3.4   1.5   0.2       Cured
## 3    6.5  3.2   5.1   2.0        Dead
## 4    6.2  2.9   4.3   1.3  Recurrence
## 5    6.7  3.3   5.7   2.5        Dead
## 6    7.9  3.8   6.4   2.0        Dead
## 7    5.6  2.7   4.2   1.3  Recurrence
## 8    5.6  3.0   4.5   1.5  Recurrence
## 9    6.2  2.2   4.5   1.5  Recurrence
## 10   4.4  3.2   1.3   0.2       Cured
## 11   6.4  2.8   5.6   2.1        Dead
## 12   5.6  2.9   3.6   1.3  Recurrence
## 13   5.2  3.5   1.5   0.2       Cured
## 14   6.0  3.0   4.8   1.8        Dead
## 15   5.0  3.2   1.2   0.2       Cured
## 16   6.1  2.6   5.6   1.4        Dead
## 17   6.1  2.8   4.7   1.2  Recurrence
## 18   6.3  2.5   4.9   1.5  Recurrence
## 19   6.3  2.7   4.9   1.8        Dead
## 20   5.5  2.3   4.0   1.3  Recurrence
## 21   5.2  4.1   1.5   0.1       Cured
## 22   5.8  2.7   5.1   1.9        Dead
## 23   7.7  3.0   6.1   2.3        Dead
## 24   5.7  2.8   4.5   1.3  Recurrence
## 25   5.4  3.0   4.5   1.5  Recurrence
## 26   5.0  3.4   1.5   0.2       Cured
## 27   5.7  2.8   4.1   1.3        Dead
## 28   5.1  3.8   1.5   0.3       Cured
## 29   4.8  3.4   1.6   0.2       Cured
## 30   5.7  3.0   4.2   1.2  Recurrence
## 31   7.7  2.6   6.9   2.3        Dead
## 32   5.4  3.9   1.3   0.4       Cured
## 33   4.9  2.4   3.3   1.0  Recurrence
## 34   5.0  3.5   1.3   0.3       Cured
## 35   6.7  3.1   5.6   2.4        Dead
## 36   7.2  3.2   6.0   1.8        Dead
## 37   6.1  3.0   4.6   1.4  Recurrence
## 38   6.4  2.8   5.6   2.2        Dead
## 39   5.7  3.8   1.7   0.3       Cured
## 40   4.5  2.3   1.3   0.3       Cured
## 41   6.8  2.8   4.8   1.4  Recurrence
## 42   4.6  3.1   1.5   0.2       Cured
## 43   6.3  3.4   5.6   2.4        Dead
## 44   7.3  2.9   6.3   1.8        Dead]
```

Calculate gini index for a given split:

The Gini index measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen. Gini index is between 0 and 1.

```
def gini_index(data, target_col):
    elements, counts = np.unique(data[target_col], return_counts = True)
    total_counts = sum(counts)
    sum_prob = 0.0
    for i in range (elements.size):
        prob_i = counts[i] / total_counts
        sum_prob = sum_prob + prob_i * prob_i

    gini_index= 1 - sum_prob
    return gini_index
```

Information gain

This function measures the reduction of the Gini index (a measure of data impurity) by dividing the dataset into two parts based on a specific feature and a threshold value. Information gain is used in decision tree algorithms to select the best feature and threshold value to split the data.

```
def information_gain(data, target_col, threshold, target_class = "Results"):
    total_gini_index = gini_index(data, "Results")
    data_left = data[data[target_col] < threshold]
    data_right = data[data[target_col] >= threshold]
    gini_index_after_split = data_left.shape[0]/ data.shape[0] * gini_index(data_left, "Results") + data_right.shape[0]/data.shape[0] * gini_index(data_right, "Results")
    info_gain = total_gini_index - gini_index_after_split
    return info_gain
```

Establish optimal splits based on the best features, best cutoffs, and best information gains

```
def selectBestFeatureAndCutoff(data, target_class = "Results"):
    featureList = list(data)[0:4]
    best_feature = "None"
    best_cutoff = 0.0
    best_info_gain = 0.0
    for feature in featureList:
        max_value = data[feature].max()
        min_value = data[feature].min()
        for cutoff in np.arange(min_value, max_value, 0.1):
            if best_info_gain < information_gain(data, feature, cutoff):
                best_info_gain = information_gain(data, feature, cutoff)
                best_cutoff = cutoff
                best_feature = feature

    return [best_feature, best_cutoff, best_info_gain]
```

Define the decision tree root (ie the first node), create the associated recursive splitting function, and create the associated prediction function

```
class Node:
    def __init__(self, feature, cut_off, label = None, is_leaf = False):
        self.feature = feature
        self.cut_off = cut_off
        self.left_child = None
        self.right_child = None
        self.is_leaf = is_leaf
        self.label = label
        #print("node's label: ")
        #print(self.label)

class DTree:
    # method to train a decision tree
    def train(self, data):
        self.root = self.build_tree(data)

    # method to build decision tree
    def build_tree(self, data):
        best_feature, best_cutoff, best_info_gain = selectBestFeatureAndCutoff(data)
        # if all data has the same label , we are at a leaf node
        if len(np.unique(data["Results"])) == 1:
            #print(data["variety"].iloc[0])
            return Node(best_feature, best_cutoff, data["Results"].iloc[0], True)

        # if we are not the leaf
        # first lets split data
        data_left = data[data[best_feature] < best_cutoff]
        data_right = data[data[best_feature] >= best_cutoff]

        #build current node
        current_node = Node(best_feature, best_cutoff)
        #add left node
        current_node.left_child = self.build_tree(data_left)
        #add right node
        current_node.right_child = self.build_tree(data_right)

        return current_node
 # Make a prediction with a decision tree
    def predict(self, data):
        current_node = self.root
        while(True):

            # if we are at the leaf node , return label
            if current_node.is_leaf == True:
                return current_node.label
            # otherwise we need figure out where to go next
            feature = current_node.feature
            cutoff = current_node.cut_off
            if data[feature]  < cutoff:
                current_node = current_node.left_child
            else:
                current_node = current_node.right_child
```

Train the decision tree

```
d_tree = DTree()
training_data = testtrain[0]
d_tree.train(training_data)
```

Define the confusion matrix

```
def print_ConfusionMatrix(result):
    count_SS = result[0]
    count_SVi = result[1]
    count_SVe = result[2]
    count_ViVi = result[3]
    count_ViVe = result[4]
    count_ViS = result[5]
    count_VeVe = result[6]
    count_VeVi = result[7]
    count_VeS = result[8]
    count_total_T =  result[9]
    count_total_F =  result[10]

    print ("True - Cured, Predicted - Cured : count_SS = ",  count_SS)
    print ("True - Cured, Predicted - Recurrence: count_SVi = ",  count_SVi)
    print ("True - Cured, Predicted - Dead: count_SVe = ",  count_SVe)

    print ("True - Dead, Predicted - Dead: count_ViVi = ",  count_ViVi)
    print ("True - Dead, Predicted - Recurrence: count_ViVe = ",  count_ViVe)
    print ("True - Dead, Predicted - Cured: Cured = ",  count_ViS)

    print ("True - Recurrence, Predicted - Recurrence: count_VeVe = ",  count_VeVe)
    print ("True - Recurrence, Predicted - Dead: count_VeVi = ",  count_VeVi)
    print ("True - Recurrence, Predicted - Cured:count_VeS = ",  count_VeS)

    print ("count_total_T = ",  count_total_T)
    print ("count_total_F = ",  count_total_F)

    print ("1) count_SS / (count_SS + count_ViS + count_VeS) = ", count_SS / (count_SS + count_ViS + count_VeS))
    if (count_SS + count_ViS + count_VeS)!=0:
      count_SS_ratio=count_SS / (count_SS + count_ViS + count_VeS)
    else:
      count_SS_ratio=0

    print ("2) count_SVi / (count_SVi + count_ViVi + count_VeVi) = ", count_SVi / (count_SVi + count_ViVi + count_VeVi))
    print ("3) count_SVe / (count_SVe + count_ViVe + count_VeVe) = ", count_SVe / (count_SVe + count_ViVe + count_VeVe))

    print ("4) count_ViS / (count_SS + count_ViS + count_VeS) = ", count_ViS / (count_SS + count_ViS + count_VeS))
    print ("5) count_ViVi / (count_SVi + count_ViVi + count_VeVi) = ", count_ViVi / (count_SVi + count_ViVi + count_VeVi))
    print ("6) count_ViVe / (count_SVe + count_ViVe + count_VeVe) = ", count_ViVe / (count_SVe + count_ViVe + count_VeVe))

    print ("7) count_VeS / (count_SS + count_ViS + count_VeS) = ", count_VeS / (count_SS + count_ViS + count_VeS))
    print ("8) count_VeVi / (count_SVi + count_ViVi + count_VeVi) = ", count_VeVi / (count_SVi + count_ViVi + count_VeVi))
    print ("9) count_VeVe / (count_SVe + count_ViVe + count_VeVe) = ", count_VeVe / (count_SVe + count_ViVe + count_VeVe))

    data = {"predict\Observe": ["Cured (predict)", "Recurrence (predict)", "Deceased (predict)"],
            "Cured (observed)": [ count_SS_ratio, count_SVi / (count_SVi + count_ViVi + count_VeVi), count_SVe / (count_SVe + count_ViVe + count_VeVe)],
            "Recurrence (observed)": [count_ViS / (count_SS + count_ViS + count_VeS), count_ViVi / (count_SVi + count_ViVi + count_VeVi), count_ViVe / (count_SVe + count_ViVe + count_VeVe)],
            "Deceased (observed)": [count_VeS / (count_SS + count_ViS + count_VeS), count_VeVi / (count_SVi + count_ViVi + count_VeVi), count_VeVe / (count_SVe + count_ViVe + count_VeVe)]
            }

    output = pd.DataFrame(data, columns = ["predict\Observe", "Cured (observed)", "Recurrence (observed)", "Deceased (observed)"])
    return output
```

Create the confusion matrix

```
def predict_batch(data):
    d_tree = DTree()
    d_tree.train(training_data)
    count_SS = 0
    count_SVi = 0
    count_SVe = 0
    count_ViVi = 0
    count_ViS = 0
    count_ViVe = 0
    count_VeVe = 0
    count_VeS = 0
    count_VeVi = 0
    count_total_T = 0
    count_total_F  = 0

    for i in range (data.shape[0]):
        instance = data.iloc[i]
        true_label = instance["Results"]
        predict_label = d_tree.predict(data.iloc[i])
        print (i, ") true_label  = ", true_label , "predict_label  = ", predict_label )
        if true_label == predict_label:
            count_total_T = count_total_T + 1
            if true_label == "Cured":
                count_SS = count_SS + 1
            elif true_label == "Dead":
                count_ViVi = count_ViVi + 1
            elif true_label == "Recurrence":
                count_VeVe = count_VeVe + 1
        else:
            count_total_F = count_total_F + 1
            if true_label == "Cured" and predict_label == "Recurrence":
                count_SVi = count_SVi + 1
            elif true_label == "Cured" and predict_label == "Dead":
                count_SVe = count_SVe + 1
            elif true_label == "Dead" and predict_label == "Recurrence":
                count_VeVi = count_VeVi + 1
            elif true_label == "Dead" and predict_label == "Cured":
                count_VeS = count_VeS + 1
            elif true_label == "Recurrence" and predict_label == "Dead":
                count_ViVe = count_ViVe + 1
            elif true_label == "Recurrence" and predict_label == "Cured":
                count_ViS = count_ViS + 1

    return [count_SS, count_SVi, count_SVe, count_ViVi, count_ViVe, count_ViS, count_VeVe, count_VeVi, count_VeS, count_total_T, count_total_F]
```

Look at the confusion matrix for training data

```
training_data = testtrain[0]
print ("training_data = ", training_data)
```

```
## training_data =       ESR1  PGR  BCL2  NAT1      Results
## 0     5.8  2.6   4.0   1.2  Recurrence
## 1     5.0  2.0   3.5   1.0  Recurrence
## 2     5.4  3.4   1.5   0.4       Cured
## 3     6.7  3.1   4.7   1.5  Recurrence
## 4     5.5  2.6   4.4   1.2  Recurrence
## ..    ...  ...   ...   ...         ...
## 100   4.9  3.1   1.5   0.1       Cured
## 101   6.1  2.8   4.0   1.3  Recurrence
## 102   4.8  3.0   1.4   0.1       Cured
## 103   5.2  2.7   3.9   1.4  Recurrence
## 104   6.5  3.0   5.8   2.2        Dead
##
## [105 rows x 5 columns]
```

```
predict_batch_results=predict_batch(training_data)
```

```
## 0 ) true_label  =  Recurrence predict_label  =  Recurrence
## 1 ) true_label  =  Recurrence predict_label  =  Recurrence
## 2 ) true_label  =  Cured predict_label  =  Cured
## 3 ) true_label  =  Recurrence predict_label  =  Recurrence
## 4 ) true_label  =  Recurrence predict_label  =  Recurrence
## 5 ) true_label  =  Cured predict_label  =  Cured
## 6 ) true_label  =  Recurrence predict_label  =  Recurrence
## 7 ) true_label = Dead predict_label = Dead
## 8 ) true_label = Dead predict_label = Dead
## 9 ) true_label = Dead predict_label = Dead
## 10 ) true_label = Cured predict_label = Cured
## 11 ) true_label = Recurrence predict_label = Recurrence
## 12 ) true_label = Recurrence predict_label = Recurrence
## 13 ) true_label = Dead predict_label = Dead
## 14 ) true_label = Cured predict_label = Cured
## 15 ) true_label = Dead predict_label = Dead
## 16 ) true_label = Cured predict_label = Cured
## 17 ) true_label = Dead predict_label = Dead
## 18 ) true_label = Dead predict_label = Dead
## 19 ) true_label = Cured predict_label = Cured
## 20 ) true_label = Dead predict_label = Dead
## 21 ) true_label = Dead predict_label = Dead
## 22 ) true_label = Cured predict_label = Cured
## 23 ) true_label = Recurrence predict_label = Recurrence
## 24 ) true_label = Dead predict_label = Dead
## 25 ) true_label = Dead predict_label = Dead
## 26 ) true_label = Cured predict_label = Cured
## 27 ) true_label = Dead predict_label = Dead
## 28 ) true_label = Dead predict_label = Dead
## 29 ) true_label = Cured predict_label = Cured
## 30 ) true_label = Dead predict_label = Dead
## 31 ) true_label = Recurrence predict_label = Recurrence
## 32 ) true_label = Cured predict_label = Cured
## 33 ) true_label = Cured predict_label = Cured
## 34 ) true_label = Recurrence predict_label = Recurrence
## 35 ) true_label = Cured predict_label = Cured
## 36 ) true_label = Dead predict_label = Dead
## 37 ) true_label = Recurrence predict_label = Recurrence
## 38 ) true_label = Cured predict_label = Cured
## 39 ) true_label = Dead predict_label = Dead
## 40 ) true_label = Dead predict_label = Dead
## 41 ) true_label = Recurrence predict_label = Recurrence
## 42 ) true_label = Cured predict_label = Cured
## 43 ) true_label = Recurrence predict_label = Recurrence
## 44 ) true_label = Recurrence predict_label = Recurrence
## 45 ) true_label = Dead predict_label = Dead
## 46 ) true_label = Cured predict_label = Cured
## 47 ) true_label = Dead predict_label = Dead
## 48 ) true_label = Dead predict_label = Dead
## 49 ) true_label = Recurrence predict_label = Recurrence
## 50 ) true_label = Dead predict_label = Dead
## 51 ) true_label = Cured predict_label = Cured
## 52 ) true_label = Cured predict_label = Cured
## 53 ) true_label = Recurrence predict_label = Recurrence
## 54 ) true_label = Dead predict_label = Dead
## 55 ) true_label = Cured predict_label = Cured
## 56 ) true_label = Recurrence predict_label = Recurrence
## 57 ) true_label = Recurrence predict_label = Recurrence
## 58 ) true_label = Cured predict_label = Cured
## 59 ) true_label = Cured predict_label = Cured
## 60 ) true_label = Recurrence predict_label = Recurrence
## 61 ) true_label = Cured predict_label = Cured
## 62 ) true_label = Cured predict_label = Cured
## 63 ) true_label = Cured predict_label = Cured
## 64 ) true_label = Cured predict_label = Cured
## 65 ) true_label = Cured predict_label = Cured
## 66 ) true_label = Recurrence predict_label = Recurrence
## 67 ) true_label = Recurrence predict_label = Recurrence
## 68 ) true_label = Cured predict_label = Cured
## 69 ) true_label = Recurrence predict_label = Recurrence
## 70 ) true_label = Recurrence predict_label = Recurrence
## 71 ) true_label = Cured predict_label = Cured
## 72 ) true_label = Cured predict_label = Cured
## 73 ) true_label = Recurrence predict_label = Recurrence
## 74 ) true_label = Dead predict_label = Dead
## 75 ) true_label = Cured predict_label = Cured
## 76 ) true_label = Dead predict_label = Dead
## 77 ) true_label = Recurrence predict_label = Recurrence
## 78 ) true_label = Dead predict_label = Dead
## 79 ) true_label = Dead predict_label = Dead
## 80 ) true_label = Dead predict_label = Dead
## 81 ) true_label = Recurrence predict_label = Recurrence
## 82 ) true_label = Recurrence predict_label = Recurrence
## 83 ) true_label = Dead predict_label = Dead
## 84 ) true_label = Recurrence predict_label = Recurrence
## 85 ) true_label  =  Recurrence predict_label  =  Recurrence
## 86 ) true_label  =  Dead predict_label  =  Dead
## 87 ) true_label  =  Dead predict_label  =  Dead
## 88 ) true_label  =  Recurrence predict_label  =  Recurrence
## 89 ) true_label  =  Cured predict_label  =  Cured
## 90 ) true_label  =  Recurrence predict_label  =  Recurrence
## 91 ) true_label  =  Cured predict_label  =  Cured
## 92 ) true_label  =  Cured predict_label  =  Cured
## 93 ) true_label  =  Cured predict_label  =  Cured
## 94 ) true_label  =  Dead predict_label  =  Dead
## 95 ) true_label  =  Dead predict_label  =  Dead
## 96 ) true_label  =  Cured predict_label  =  Cured
## 97 ) true_label  =  Recurrence predict_label  =  Recurrence
## 98 ) true_label  =  Dead predict_label  =  Dead
## 99 ) true_label  =  Cured predict_label  =  Cured
## 100 ) true_label  =  Cured predict_label  =  Cured
## 101 ) true_label  =  Recurrence predict_label  =  Recurrence
## 102 ) true_label  =  Cured predict_label  =  Cured
## 103 ) true_label  =  Recurrence predict_label  =  Recurrence
## 104 ) true_label  =  Dead predict_label  =  Dead
```

```
print ("predict_batch_results = ", predict_batch_results)
```

```
## predict_batch_results =  [37, 0, 0, 34, 0, 0, 34, 0, 0, 105, 0]
```

```
print_ConfusionMatrix(predict_batch(training_data))
```

```
## 0 ) true_label  =  Recurrence predict_label  =  Recurrence
## 1 ) true_label  =  Recurrence predict_label  =  Recurrence
## 2 ) true_label  =  Cured predict_label  =  Cured
## 3 ) true_label  =  Recurrence predict_label  =  Recurrence
## 4 ) true_label  =  Recurrence predict_label  =  Recurrence
## 5 ) true_label  =  Cured predict_label  =  Cured
## 6 ) true_label  =  Recurrence predict_label  =  Recurrence
## 7 ) true_label  =  Dead predict_label  =  Dead
## 8 ) true_label  =  Dead predict_label  =  Dead
## 9 ) true_label  =  Dead predict_label  =  Dead
## 10 ) true_label  =  Cured predict_label  =  Cured
## 11 ) true_label  =  Recurrence predict_label  =  Recurrence
## 12 ) true_label  =  Recurrence predict_label  =  Recurrence
## 13 ) true_label  =  Dead predict_label  =  Dead
## 14 ) true_label  =  Cured predict_label  =  Cured
## 15 ) true_label  =  Dead predict_label  =  Dead
## 16 ) true_label  =  Cured predict_label  =  Cured
## 17 ) true_label  =  Dead predict_label  =  Dead
## 18 ) true_label  =  Dead predict_label  =  Dead
## 19 ) true_label  =  Cured predict_label  =  Cured
## 20 ) true_label  =  Dead predict_label  =  Dead
## 21 ) true_label  =  Dead predict_label  =  Dead
## 22 ) true_label  =  Cured predict_label  =  Cured
## 23 ) true_label  =  Recurrence predict_label  =  Recurrence
## 24 ) true_label  =  Dead predict_label  =  Dead
## 25 ) true_label  =  Dead predict_label  =  Dead
## 26 ) true_label  =  Cured predict_label  =  Cured
## 27 ) true_label  =  Dead predict_label  =  Dead
## 28 ) true_label  =  Dead predict_label  =  Dead
## 29 ) true_label  =  Cured predict_label  =  Cured
## 30 ) true_label  =  Dead predict_label  =  Dead
## 31 ) true_label  =  Recurrence predict_label  =  Recurrence
## 32 ) true_label  =  Cured predict_label  =  Cured
## 33 ) true_label  =  Cured predict_label  =  Cured
## 34 ) true_label  =  Recurrence predict_label  =  Recurrence
## 35 ) true_label  =  Cured predict_label  =  Cured
## 36 ) true_label  =  Dead predict_label  =  Dead
## 37 ) true_label  =  Recurrence predict_label  =  Recurrence
## 38 ) true_label  =  Cured predict_label  =  Cured
## 39 ) true_label  =  Dead predict_label  =  Dead
## 40 ) true_label  =  Dead predict_label  =  Dead
## 41 ) true_label  =  Recurrence predict_label  =  Recurrence
## 42 ) true_label  =  Cured predict_label  =  Cured
## 43 ) true_label  =  Recurrence predict_label  =  Recurrence
## 44 ) true_label  =  Recurrence predict_label  =  Recurrence
## 45 ) true_label  =  Dead predict_label  =  Dead
## 46 ) true_label  =  Cured predict_label  =  Cured
## 47 ) true_label  =  Dead predict_label  =  Dead
## 48 ) true_label  =  Dead predict_label  =  Dead
## 48 ) true_label  =  Dead predict_label  =  Dead
## 49 ) true_label  =  Recurrence predict_label  =  Recurrence
## 50 ) true_label  =  Dead predict_label  =  Dead
## 51 ) true_label  =  Cured predict_label  =  Cured
## 52 ) true_label  =  Cured predict_label  =  Cured
## 53 ) true_label  =  Recurrence predict_label  =  Recurrence
## 54 ) true_label  =  Dead predict_label  =  Dead
## 55 ) true_label  =  Cured predict_label  =  Cured
## 56 ) true_label  =  Recurrence predict_label  =  Recurrence
## 57 ) true_label  =  Recurrence predict_label  =  Recurrence
## 58 ) true_label  =  Cured predict_label  =  Cured
## 59 ) true_label  =  Cured predict_label  =  Cured
## 60 ) true_label  =  Recurrence predict_label  =  Recurrence
## 61 ) true_label  =  Cured predict_label  =  Cured
## 62 ) true_label  =  Cured predict_label  =  Cured
## 63 ) true_label  =  Cured predict_label  =  Cured
## 64 ) true_label  =  Cured predict_label  =  Cured
## 65 ) true_label  =  Cured predict_label  =  Cured
## 66 ) true_label  =  Recurrence predict_label  =  Recurrence
## 67 ) true_label  =  Recurrence predict_label  =  Recurrence
## 68 ) true_label  =  Cured predict_label  =  Cured
## 69 ) true_label  =  Recurrence predict_label  =  Recurrence
## 70 ) true_label  =  Recurrence predict_label  =  Recurrence
## 71 ) true_label  =  Cured predict_label  =  Cured
## 72 ) true_label  =  Cured predict_label  =  Cured
## 73 ) true_label  =  Recurrence predict_label  =  Recurrence
## 74 ) true_label  =  Dead predict_label  =  Dead
## 75 ) true_label  =  Cured predict_label  =  Cured
## 76 ) true_label  =  Dead predict_label  =  Dead
## 77 ) true_label  =  Recurrence predict_label  =  Recurrence
## 78 ) true_label  =  Dead predict_label  =  Dead
## 79 ) true_label  =  Dead predict_label  =  Dead
## 80 ) true_label  =  Dead predict_label  =  Dead
## 81 ) true_label  =  Recurrence predict_label  =  Recurrence
## 82 ) true_label  =  Recurrence predict_label  =  Recurrence
## 83 ) true_label  =  Dead predict_label  =  Dead
## 84 ) true_label  =  Recurrence predict_label  =  Recurrence
## 85 ) true_label  =  Recurrence predict_label  =  Recurrence
## 86 ) true_label  =  Dead predict_label  =  Dead
## 87 ) true_label  =  Dead predict_label  =  Dead
## 88 ) true_label  =  Recurrence predict_label  =  Recurrence
## 89 ) true_label  =  Cured predict_label  =  Cured
## 90 ) true_label  =  Recurrence predict_label  =  Recurrence
## 91 ) true_label  =  Cured predict_label  =  Cured
## 92 ) true_label  =  Cured predict_label  =  Cured
## 93 ) true_label  =  Cured predict_label  =  Cured
## 94 ) true_label  =  Dead predict_label  =  Dead
## 95 ) true_label  =  Dead predict_label  =  Dead
## 96 ) true_label  =  Cured predict_label  =  Cured
## 97 ) true_label  =  Recurrence predict_label  =  Recurrence
## 98 ) true_label  =  Dead predict_label  =  Dead
## 99 ) true_label  =  Cured predict_label  =  Cured
## 100 ) true_label  =  Cured predict_label  =  Cured
## 101 ) true_label  =  Recurrence predict_label  =  Recurrence
## 102 ) true_label  =  Cured predict_label  =  Cured
## 103 ) true_label  =  Recurrence predict_label  =  Recurrence
## 104 ) true_label  =  Dead predict_label  =  Dead
## True - Cured, Predicted - Cured : count_SS =  37
## True - Cured, Predicted - Recurrence: count_SVi =  0
## True - Cured, Predicted - Dead: count_SVe =  0
## True - Dead, Predicted - Dead: count_ViVi =  34
## True - Dead, Predicted - Recurrence: count_ViVe =  0
## True - Dead, Predicted - Cured: Cured =  0
## True - Recurrence, Predicted - Recurrence: count_VeVe =  34
## True - Recurrence, Predicted - Dead: count_VeVi =  0
## True - Recurrence, Predicted - Cured:count_VeS =  0
## count_total_T =  105
## count_total_F =  0
## 1) count_SS / (count_SS + count_ViS + count_VeS) =  1.0
## 2) count_SVi / (count_SVi + count_ViVi + count_VeVi) =  0.0
## 3) count_SVe / (count_SVe + count_ViVe + count_VeVe) =  0.0
## 4) count_ViS / (count_SS + count_ViS + count_VeS) =  0.0
## 5) count_ViVi / (count_SVi + count_ViVi + count_VeVi) =  1.0
## 6) count_ViVe / (count_SVe + count_ViVe + count_VeVe) =  0.0
## 7) count_VeS / (count_SS + count_ViS + count_VeS) =  0.0
## 8) count_VeVi / (count_SVi + count_ViVi + count_VeVi) =  0.0
## 9) count_VeVe / (count_SVe + count_ViVe + count_VeVe) =  1.0
##         predict\Observe  ...  Deceased (observed)
## 0       Cured (predict)  ...                  0.0
## 0       Cured (predict)  ...                 0.0
## 1  Recurrence (predict)  ...                 0.0
## 2    Deceased (predict)  ...                 1.0
## 
## [3 rows x 4 columns]
```

Look at the confusion matrix for testing data

```
testing_data = testtrain[1]
print_ConfusionMatrix(predict_batch(testing_data))
```

```
## 0 ) true_label  =  Recurrence predict_label  =  Recurrence
## 1 ) true_label  =  Dead predict_label  =  Dead
## 2 ) true_label  =  Cured predict_label  =  Cured
## 3 ) true_label  =  Dead predict_label  =  Dead
## 4 ) true_label  =  Recurrence predict_label  =  Recurrence
## 5 ) true_label  =  Dead predict_label  =  Dead
## 6 ) true_label  =  Dead predict_label  =  Dead
## 7 ) true_label  =  Recurrence predict_label  =  Recurrence
## 8 ) true_label  =  Recurrence predict_label  =  Recurrence
## 9 ) true_label  =  Recurrence predict_label  =  Recurrence
## 10 ) true_label  =  Cured predict_label  =  Cured
## 11 ) true_label  =  Dead predict_label  =  Dead
## 12 ) true_label  =  Recurrence predict_label  =  Recurrence
## 13 ) true_label  =  Cured predict_label  =  Cured
## 14 ) true_label  =  Dead predict_label  =  Dead
## 15 ) true_label  =  Cured predict_label  =  Cured
## 16 ) true_label  =  Dead predict_label  =  Dead
## 17 ) true_label  =  Recurrence predict_label  =  Recurrence
## 18 ) true_label  =  Recurrence predict_label  =  Dead
## 19 ) true_label  =  Dead predict_label  =  Dead
## 20 ) true_label  =  Recurrence predict_label  =  Recurrence
## 21 ) true_label  =  Cured predict_label  =  Cured
## 22 ) true_label  =  Dead predict_label  =  Dead
## 23 ) true_label  =  Dead predict_label  =  Dead
## 24 ) true_label  =  Recurrence predict_label  =  Recurrence
## 25 ) true_label  =  Recurrence predict_label  =  Recurrence
## 26 ) true_label  =  Cured predict_label  =  Cured
## 27 ) true_label  =  Dead predict_label  =  Recurrence
## 28 ) true_label  =  Cured predict_label  =  Cured
## 29 ) true_label  =  Cured predict_label  =  Cured
## 30 ) true_label  =  Recurrence predict_label  =  Recurrence
## 31 ) true_label  =  Dead predict_label  =  Dead
## 32 ) true_label  =  Cured predict_label  =  Cured
## 33 ) true_label  =  Recurrence predict_label  =  Dead
## 34 ) true_label  =  Cured predict_label  =  Cured
## 35 ) true_label  =  Dead predict_label  =  Dead
## 36 ) true_label  =  Dead predict_label  =  Dead
## 37 ) true_label  =  Recurrence predict_label  =  Recurrence
## 38 ) true_label  =  Dead predict_label  =  Dead
## 39 ) true_label  =  Cured predict_label  =  Cured
## 40 ) true_label  =  Cured predict_label  =  Cured
## 41 ) true_label  =  Recurrence predict_label  =  Recurrence
## 42 ) true_label  =  Cured predict_label  =  Cured
## 43 ) true_label  =  Dead predict_label  =  Dead
## 44 ) true_label  =  Dead predict_label  =  Dead
## True - Cured, Predicted - Cured : count_SS =  13
## True - Cured, Predicted - Recurrence: count_SVi =  0
## True - Cured, Predicted - Dead: count_SVe =  0
## True - Dead, Predicted - Dead: count_ViVi =  16
## True - Dead, Predicted - Recurrence: count_ViVe =  2
## True - Dead, Predicted - Cured: Cured =  0
## True - Recurrence, Predicted - Recurrence: count_VeVe =  13
## True - Recurrence, Predicted - Dead: count_VeVi =  1
## True - Recurrence, Predicted - Cured:count_VeS =  0
## count_total_T =  42
## count_total_F =  3
## 1) count_SS / (count_SS + count_ViS + count_VeS) =  1.0
## 2) count_SVi / (count_SVi + count_ViVi + count_VeVi) =  0.0
## 3) count_SVe / (count_SVe + count_ViVe + count_VeVe) =  0.0
## 4) count_ViS / (count_SS + count_ViS + count_VeS) =  0.0
## 5) count_ViVi / (count_SVi + count_ViVi + count_VeVi) =  0.9411764705882353
## 6) count_ViVe / (count_SVe + count_ViVe + count_VeVe) =  0.13333333333333333
## 7) count_VeS / (count_SS + count_ViS + count_VeS) =  0.0
## 8) count_VeVi / (count_SVi + count_ViVi + count_VeVi) =  0.058823529411764705
## 9) count_VeVe / (count_SVe + count_ViVe + count_VeVe) =  0.8666666666666667
##         predict\Observe  ...  Deceased (observed)
## 0       Cured (predict)  ...             0.000000
## 1  Recurrence (predict)  ...             0.058824
## 2    Deceased (predict)  ...             0.866667
## 
## [3 rows x 4 columns]
```

Make predictions

```
# method that run prediction
def predict(d_tree, ESR1, PGR, BCL2, NAT1):
    test_data = pd.Series([ESR1, PGR, BCL2, NAT1], index = ['ESR1', 'PGR', 'BCL2', 'NAT1'])
    return d_tree.predict(test_data)
```

- Predict a new ind with [ESR1=1 , PGR=1 , BCL2=1 , NAT1=1]

```
predict_features = [1, 1, 1, 1]

result_category = predict(d_tree, predict_features[0], predict_features[1], predict_features[2], predict_features[3])

print("This patient is ", result_category)
```

```
## This patient is  Cured
```

- Predict a new ind with [ESR1=1 , PGR=2 , BCL2=3 , NAT1=4]

```
predict_features = [1, 2, 3, 4]

result_category = predict(d_tree, predict_features[0], predict_features[1], predict_features[2], predict_features[3])

print("This patient is ", result_category)
```

```
## This patient is  Dead
```