Import Library

In [1]:

```
import numpy
from scipy.stats import entropy
from pandas import DataFrame, read_csv
import matplotlib.pyplot as plotLib
import pandas as pd
import pprint
```

Preparing Dataset

Load Dataset from xls

In [80]:

```
df = pd.read_csv("../sources/heart_failure_clinical_records_dataset.csv")
df = df[['anaemia', 'high_blood_pressure', 'sex', 'smoking', 'diabetes', 'DEATH_EVENT']]
print(df.describe())
df['DEATH_EVENT']= df['DEATH_EVENT'].replace([0, 1], ['none', 'heart_failure'])
df['sex']= df['sex'].replace([0, 1], ['Female', 'Male'])
df
```

```
          anaemia  high_blood_pressure         sex    smoking    diabetes  \
count  299.000000           299.000000  299.000000  299.00000  299.000000
mean     0.431438             0.351171    0.648829    0.32107    0.418060
std      0.496107             0.478136    0.478136    0.46767    0.494067
min      0.000000             0.000000    0.000000    0.00000    0.000000
25%      0.000000             0.000000    0.000000    0.00000    0.000000
50%      0.000000             0.000000    1.000000    0.00000    0.000000
75%      1.000000             1.000000    1.000000    1.00000    1.000000
max      1.000000             1.000000    1.000000    1.00000    1.000000

       DEATH_EVENT
count    299.00000
mean       0.32107
std        0.46767
min        0.00000
25%        0.00000
50%        0.00000
75%        1.00000
max        1.00000
```

Out[80]:

	anaemia	high_blood_pressure	sex	smoking	diabetes	DEATH_EVENT
0	0	1	Male	0	0	heart_failure
1	0	0	Male	0	0	heart_failure
2	0	0	Male	1	0	heart_failure
3	1	0	Male	0	0	heart_failure
4	1	0	Female	0	1	heart_failure
...	...	...	...	...	...	...
294	0	1	Male	1	1	none
295	0	0	Female	0	0	none
296	0	0	Female	0	1	none
297	0	0	Male	1	0	none

298	anaemia 0	high_blood_pressure 0	Male	smoking 1	diabetes 0	DEATH_EVENT	

299 rows × 6 columns

Get Input Attributes

In [3]:

```
selectedInput = ['anaemia', 'high_blood_pressure', 'sex', 'smoking', 'diabetes']
inputAttributes = df[selectedInput]
inputAttributes
```

Out[3]:

	anaemia	high_blood_pressure	sex	smoking	diabetes
0	0	1	1	0	0
1	0	0	1	0	0
2	0	0	1	1	0
3	1	0	1	0	0
4	1	0	0	0	1
...	...	...	...	...	...
294	0	1	1	1	1
295	0	0	0	0	0
296	0	0	0	0	1
297	0	0	1	1	0
298	0	0	1	1	0

299 rows × 5 columns

Get Input Attributes' values

In [4]:

```
anmValues = inputAttributes.anaemia.unique()
hbpValues = inputAttributes.high_blood_pressure.unique()
sexValues = inputAttributes.sex.unique()
smkValues = inputAttributes.smoking.unique()
dbtValues = inputAttributes.diabetes.unique()

print(anmValues)
print(hbpValues)
print(sexValues)
print(smkValues)
print(dbtValues)
```

```
[0 1]
[1 0]
[1 0]
[0 1]
[0 1]
```

Get target attributes

In [5]:

```
selectedTarget = ['DEATH_EVENT']
targetAttribute = df[selectedTarget]
targetAttribute
```

Out[5]:

	DEATH_EVENT
0	heart_failure
1	heart_failure
2	heart_failure
3	heart_failure
4	heart_failure
...	...
294	none
295	none
296	none
297	none
298	none

299 rows × 1 columns

Get target attributes' values

In [6]:

```
targetValues = targetAttribute.DEATH_EVENT.unique()
```

Count Instances and Target Distribution

In [7]:

```
targetDataFrame = df['DEATH_EVENT'].value_counts()
totalInstance = df['DEATH_EVENT'].value_counts().sum()
```

Preparing attributes impurity

Get Anemia and its values

In [8]:

```
getAnmYes = df.loc[df["anaemia"] == "1"]
getAnmNo = df.loc[df["anaemia"] == "0"]
```

In [9]:

```
targetAnmYes = getAnmYes.DEATH_EVENT.value_counts()
targetAnmNo = getAnmNo.DEATH_EVENT.value_counts()
```

In [10]:

```
getAnmYesSum = targetAnmYes.sum()
getAnmNoSum = targetAnmNo.sum()
```

Get High Blood Pressure and its values

In [11]:

```
getHBPYes = df.loc[df["high_blood_pressure"] == "1"]
getHBPNo = df.loc[df["high_blood_pressure"] == "0"]
```

In [12]:

```
targetHBPYes = getHBPYes.DEATH_EVENT.value_counts()
targetHBPNo = getHBPNo.DEATH_EVENT.value_counts()
```

In [13]:

```
getHBPYesSum = targetHBPYes.sum()
getHBPNoSum = targetHBPNo.sum()
```

Get Sex and its values

In [81]:

```
getMale = df.loc[df["sex"] == "Male"]
getFemale = df.loc[df["sex"] == "Female"]
```

In [82]:

```
targetMale = getMale.DEATH_EVENT.value_counts()
targetFemale = getFemale.DEATH_EVENT.value_counts()
```

In [83]:

```
getMaleSum = targetMale.sum()
getFemaleSum = targetFemale.sum()
```

Get Smoking and its values

In [17]:

```
getSmkNo = df.loc[df['smoking']=="0"]
getSmkYes = df.loc[df['smoking']=="1"]
```

In [18]:

```
targetSmkNo = getSmkNo.DEATH_EVENT.value_counts()
targetSmkYes = getSmkYes.DEATH_EVENT.value_counts()
```

In [19]:

```
getSmkNoSum = targetSmkNo.sum()
getSmkYesSum = targetSmkYes.sum()
```

Get Diabetes and its values

In [20]:

```
getDbtNo = df.loc[df['diabetes']=="0"]
getDbtYes = df.loc[df['diabetes']=="1"]
```

In [21]:

```
targetDbtNo = getDbtNo.DEATH_EVENT.value_counts()
targetDbtYes = getDbtYes.DEATH_EVENT.value_counts()
```

In [22]:

```
getDbtNoSum = targetDbtNo.sum()
getDbtYesSum = targetDbtYes.sum()
```

Let's Loop it

In [23]:

```
targetDataFrame
```

Out[23]:

```
none             203
heart_failure     96
Name: DEATH_EVENT, dtype: int64
```

Gini Function

In [24]:

```
def Gini(p, q):
    result = 1 - ((p**2)+(q**2))
    return result
```

Calculate DF's base entropy and gini

In [26]:

```
p = targetDataFrame[0]/totalInstance
q = targetDataFrame[1]/totalInstance

baseEntropy = entropy([p,q], base=2)
baseGini = Gini(p, q)

print("Base Entropy:", baseEntropy)
print("Base Gini:", baseGini)
```

```
Base Entropy: 0.9055415027672631
Base Gini: 0.43596827775975666
```

Loop each attribute and loop each value

In [27]:

```
for attribute in selectedInput:
    print("Attribute: ", attribute)
    attributeValues = df[attribute].unique()
    #
    sumEntropy = 0
    sumGini = 0
    for value in attributeValues:
        subDataFrame = df.loc[df[attribute]==value]
        sumDataFrame = subDataFrame[selectedTarget].value_counts().sum()
        #
        print("Value: ", value, " , Total: ", sumDataFrame, " of ", totalInstance, " Ins
tances")
        print(subDataFrame[selectedTarget].value_counts().to_frame())
        #
        totalIndex = subDataFrame[selectedTarget].value_counts().count()
        p = subDataFrame[selectedTarget].value_counts()[0]/sumDataFrame
        if(totalIndex == 1):
            q = 0
        else:
            q = subDataFrame[selectedTarget].value_counts()[1]/sumDataFrame
        #
        valueEntropy = (sumDataFrame/totalInstance)*(entropy([p,q], base=2))
        sumEntropy += valueEntropy
```

```
Attribute:  anaemia
Value:  0  , Total:  170  of  299  Instances
                0
DEATH_EVENT
none          120
```

```
heart_failure   50
Value:  1   , Total:  129  of  299  Instances
                0
DEATH_EVENT
none           83
heart_failure  46
Attribute:  high_blood_pressure
Value:  1   , Total:  105  of  299  Instances
                0
DEATH_EVENT
none           66
heart_failure  39
Value:  0   , Total:  194  of  299  Instances
                0
DEATH_EVENT
none           137
heart_failure   57
Attribute:  sex
Value:  1   , Total:  194  of  299  Instances
                0
DEATH_EVENT
none           132
heart_failure   62
Value:  0   , Total:  105  of  299  Instances
                0
DEATH_EVENT
none           71
heart_failure  34
Attribute:  smoking
Value:  0   , Total:  203  of  299  Instances
                0
DEATH_EVENT
none           137
heart_failure   66
Value:  1   , Total:  96  of  299  Instances
                0
DEATH_EVENT
none           66
heart_failure  30
Attribute:  diabetes
Value:  0   , Total:  174  of  299  Instances
                0
DEATH_EVENT
none           118
heart_failure   56
Value:  1   , Total:  125  of  299  Instances
                0
DEATH_EVENT
none           85
heart_failure  40
```

Let's recursive it!

Function to calculate Parent's entropy

In [28]:

```
def getParentEntropy(df, target):
    targetDataFrame = df[target].value_counts()
    totalInstance = df[target].value_counts().sum()
    #
    p = targetDataFrame[0]/totalInstance
    q = targetDataFrame[1]/totalInstance
    #
    baseEntropy = entropy([p,q], base=2)
    baseGini = Gini(p, q)
    #
    result = baseEntropy
```

```
    return result
```

Function to Calculate Children's entropy

In [29]:

```
def getChildEntropy(df, target, attribute):
    totalInstance = df[target].value_counts().sum()
    attributeValues = df[attribute].unique()
    #
    sumEntropy = 0
    sumGini = 0
    for value in attributeValues:
        subDataFrame = df.loc[df[attribute]==value]
        sumDataFrame = subDataFrame[selectedTarget].value_counts().sum()
        #
        valueEntropy = 0
        valueGini = 0
        #
        totalIndex = subDataFrame[selectedTarget].value_counts().count()
        if(totalIndex <1):
            p=0
            q=0
        else:
            p = subDataFrame[selectedTarget].value_counts()[0]/sumDataFrame
            if(totalIndex == 1):
                q = 0
            else:
                q = subDataFrame[selectedTarget].value_counts()[1]/sumDataFrame
            #
            valueEntropy = (sumDataFrame/totalInstance)*(entropy([p,q], base=2))
            valueGini = (sumDataFrame/totalInstance)*(Gini(p,q))
        #
        sumEntropy += valueEntropy
        sumGini += valueGini
    result = sumEntropy
    return abs(result)
```

Function to find best attribute

In [30]:

```
def findBestAttribute(df, target):
    gainList = []
    #
    parentEntropy = getParentEntropy(df, target)
    for attribute in df.keys()[:-1]:
        childEntropy = getChildEntropy(df, target, attribute)
        gain = parentEntropy - childEntropy
        gainList.append(gain)

    result = df.keys()[numpy.argmax(gainList)]
    return result
```

Function to Split Dataset

In [31]:

```
def getSubBranch(df, attribute, value):
    return df[df[attribute]==value].reset_index(drop=True)
```

Function Recursive ID3

In [78]:

```
def ID3(df, tree=None, counter=0):
    selectedTarget = ['DEATH_EVENT']
    bestAttribute = findBestAttribute(df, selectedTarget)
    bestAttValues = df[bestAttribute].unique()
    if tree is None:
        tree = {}
        tree[bestAttribute] = {}
    for value in bestAttValues:
        nextBranch = getSubBranch(df, bestAttribute, value)
        nextBranchValue, nextBranchCounts = numpy.unique(nextBranch[selectedTarget], ret
urn_counts=True)
#         print(int(numpy.where(nextBranchCounts==max(nextBranchCounts))[0]))
        if len(nextBranchCounts)==1:
            tree[bestAttribute][value] = nextBranchValue[0]
        elif(counter<5):
            tree[bestAttribute][value] = ID3(nextBranch, counter=counter+1)
        else:
            valueIndex = numpy.where(nextBranchCounts==max(nextBranchCounts))[0][0]
            tree[bestAttribute][value] = nextBranchValue[valueIndex]

    result = tree
    return result
```

Let's Run Our ID3

In [84]:

```
myTree = ID3(df)
pprint.pprint(myTree)
```

```
{'high_blood_pressure': {0: {'smoking': {0: {'diabetes': {0: {'sex': {'Female': {'anaemia
': {0: {'anaemia': {0: 'none'}},

1: {'anaemia': {1: 'none'}}}},
                                                                    'Male': {'anaemia
': {0: {'anaemia': {0: 'none'}},

1: {'anaemia': {1: 'heart_failure'}}}}}},
                                                          1: {'anaemia': {0: {'sex': {'F
emale': {'anaemia': {0: 'none'}},
                                                                                      '
Male': {'anaemia': {0: 'none'}}}},
                                                                          1: {'sex': {'
Female': {'anaemia': {1: 'none'}},
                                                                                      '
Male': {'anaemia': {1: 'none'}}}}}}}},
                                          1: {'sex': {'Female': 'heart_failure',
                                                      'Male': {'diabetes': {0: {'anaemia'
: {0: {'anaemia': {0: 'none'}},

1: {'anaemia': {1: 'none'}}}},
                                                                           1: {'anaemia
': {0: {'anaemia': {0: 'none'}},

1: {'anaemia': {1: 'none'}}}}}}}}}},
                            1: {'smoking': {0: {'diabetes': {0: {'anaemia': {0: {'sex': {'Fe
male': {'anaemia': {0: 'none'}},
                                                                                      '
Male': {'anaemia': {0: 'none'}}}},
                                                                          1: {'sex': {'
Female': {'anaemia': {1: 'none'}},
                                                                                      '
Male': {'anaemia': {1: 'none'}}}}}},
                                                                1: {'sex': {'Female': {'anaemi
a': {0: {'anaemia': {0: 'none'}},

1: {'anaemia': {1: 'heart_failure'}}}},
                                                                    'Male': {'anaemia
': {0: {'anaemia': {0: 'none'}},
```

```
1: 'none'}}}}}},
                                                1: {'sex': {'Female': {'anaemia': {0: {'diabete
s': {0: 'none',
1: 'heart_failure'}},
                                                                                1: 'heart_f
ailure'}},
                                                          'Male': {'diabetes': {0: {'anaemia'
: {0: {'anaemia': {0: 'none'}},
1: {'anaemia': {1: 'heart_failure'}}}},
                                                                                1: {'anaemia
': {0: {'anaemia': {0: 'heart_failure'}},
1: 'none'}}}}}}}}}}
```