Importing the required packages

In [3]:

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

Function importing Dataset

In [14]:

```
def importdata():
    balance_data = pd.read_csv('final.csv', sep= ',', header = 0)

    # Printing the dataswet shape
    print ("Dataset Lenght: ", len(balance_data))
    print ("Dataset Shape: ", balance_data.shape)

    # Printing the dataset obseravtions
    return balance_data
```

Function to split the dataset

In [15]:

```
def splitdataset(balance_data):

    # Seperating the target variable
    X = balance_data.values[:, 0:24]
    Y = balance_data.values[:, -1]

    #print(X)
    #print(Y)

    # Spliting the dataset into train and test
    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.3, random_state = 100)

    return X, Y, X_train, X_test, y_train, y_test
```

Function to perform training with giniIndex.

In [16]:

```
def train_using_gini(X_train, X_test, y_train):

    # Creating the classifier object
    clf_gini = DecisionTreeClassifier(criterion = "gini",
            random_state = 100,max_depth=3, min_samples_leaf=5)

    # Performing training
    clf_gini.fit(X_train, y_train)
    return clf_gini
```

Function to perform training with entropy.

In [17]:

```
def tarin_using_entropy(X_train, X_test, y_train):

    # Decision tree with entropy
    clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state = 100,
            max_depth = 3, min_samples_leaf = 5)

    # Performing training
    clf_entropy.fit(X_train, y_train)
    return clf_entropy
```

Function to make predictions

In [18]:

```
def prediction(X_test, clf_object):

    # Predicton on test with giniIndex
    y_pred = clf_object.predict(X_test)
    return y_pred
```

Function to calculate accuracy

In [19]:

```
def cal_accuracy(y_test, y_pred):

    print("Confusion Matrix: \n",
    confusion_matrix(y_test,y_pred))

    print ("Accuracy : ",
    accuracy_score(y_test,y_pred)*100)

    print("Report : \n",
    classification_report(y_test, y_pred))
```

Main code

In [20]:

```
def main():

    # Building Phase
    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

    # Operational Phase
    print("Results Using Gini Index:")

    # Prediction using gini
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)

    print("Results Using Entropy:")
    # Prediction using entropy
    y_pred_entropy = prediction(X_test, clf_entropy)
    cal_accuracy(y_test, y_pred_entropy)
```

Calling main function

In [21]:

```
if __name__=="__main__":
    main()
```

```
Dataset Lenght:  400
Dataset Shape:  (400, 25)
Results Using Gini Index:
Confusion Matrix:
 [[77  3]
 [ 1 39]]
Accuracy :  96.66666666666667
Report :
              precision    recall  f1-score   support

         ckd       0.99      0.96      0.97        80
      notckd       0.93      0.97      0.95        40

   micro avg       0.97      0.97      0.97       120
   macro avg       0.96      0.97      0.96       120
weighted avg       0.97      0.97      0.97       120

Results Using Entropy:
Confusion Matrix:
 [[77  3]
 [ 1 39]]
Accuracy :  96.66666666666667
Report :
              precision    recall  f1-score   support

         ckd       0.99      0.96      0.97        80
      notckd       0.93      0.97      0.95        40

   micro avg       0.97      0.97      0.97       120
   macro avg       0.96      0.97      0.96       120
weighted avg       0.97      0.97      0.97       120
```

In [ ]: