In [2]:

```
import numpy as np
from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from tqdm import tqdm
```

In [3]:

```
boston = load_boston()
x=boston.data #independent variables
y=boston.target #target variable
```

In [5]:

```
#===================================================================================================
=======================#
#                                                    TASK-1
#
#===================================================================================================
=======================#
#===================================================================================================
=======================#
#    Step-1
#
#===================================================================================================
=======================#
#creating 30 samples from the whole boston data points.
Y_MSE_LIST=[]
Y_OOB_LIST=[]
Y_MSE_LIST_SKT=[]
for T in tqdm(range(35)):
    T=T+1
    col=np.arange(13)
    x_num=np.arange(len(x))
    y_num=np.arange(len(y))
    col_set=[]
    X_train=[]
    for i in range(30):
        i=i+1
    #randomly choosing different feature for each sample
        col_list=np.random.choice(col,6,replace=False)
        col_sort=np.sort(col_list)
        col_set.append(col_sort)

        X_set_60=np.random.choice(x_num,304,replace=False)#selecting 60% data
        X_set_40=np.random.choice(X_set_60,202,replace=False)#selecting 40%data from 60%data
    #addiing up the 60% and 40% values
        X_set_final=np.hstack((X_set_60,X_set_60))
        X_set_final=np.sort(X_set_final)
    #Corpus of 30 samples
        X_train.append(X_set_final)
#===================================================================================================
=======================#
#    Step-2
#
#===================================================================================================
=======================#

    from sklearn.tree import DecisionTreeRegressor
    clf=DecisionTreeRegressor()
    y_pred=[]
    y_pred_test=[]

    #y_pred corpus
    for i in range(30):
        y_total_pred=[]
        clf.fit(x[:,col_set[i]][X_train[i]], y[X_train[i]])
        y_pred.append(clf.predict(x[:,col_set[i]]))
    #computing Average y_pre value
    y_sum=y_pred[0]
    for j in range(len(y_pred)):
        y_sum=+np.add(y_sum,y_pred[j])
    y_pred_avg=np.true_divide(y_sum, 30)
    #computing MSE:
    from sklearn.metrics import mean_squared_error as mse
    y_mse_scikit=mse(y_pred_avg,y)#using scikit
    #print(y_mse_scikit)
    Y_MSE_LIST_SKT.append(round(y_mse_scikit,2))
    y_mse=np.subtract(y,y_pred_avg)#y-y_pred
    y_mse_sqr=np.square(y_mse)#squaring the diff
    y_mse_sum=0
    for i in range(len(y_mse_sqr)):
        y_mse_sum=y_mse_sum+y_mse_sqr[i]
    y_mse_final=y_mse_sum/len(y_mse_sqr)
    #print(y_mse_final)
    Y_MSE_LIST.append(round(y_mse_final,2))

#=================================================================================================
=======================#
#    Step-3
#
#=================================================================================================
=======================#

    y_pred_oob=[]
    countp=[]

    X_trains=np.asarray(X_train)
    for i in range(len(x)):
        count=0
        sum1=0
        for j in range(30):
            #print(i,j)
            if i not in X_train[j]:
                #print(i,j)
                count=count+1
                #print(count)
                clf.fit(x[:,col_set[j]][X_train[j]], y[X_train[j]])
                sum1=sum1+clf.predict((x[:,col_set[0]][0].reshape(1,-1)))
        y_pred_oob.extend(sum1/count)

#computing OOB SCORE:
    y_oob=np.subtract(y,y_pred_oob)#y-y_pred_oob
    y_oob_sqr=np.square(y_oob)#squaring the diff
    y_oob_sum=0
#print(y_oob_sqr)
    for i in range(len(y_oob_sqr)):
        y_oob_sum=y_oob_sum+y_oob_sqr[i]
    y_oob_final=y_oob_sum/(len(y_oob_sqr))
    #print((y_oob_final))
    Y_OOB_LIST.append(round(y_oob_final,2))
```

```
100%|██████████████████████████████████████████████████████████████████████████████████| 35/35
[10:41<00:00, 18.95s/it]
```

In [6]:

```
print('y_mse values through sklearn Y_MSE_LIST_SKT:\n',Y_MSE_LIST_SKT)
print('\nMSE values calculated:\n',Y_MSE_LIST)
print('\nOOB_values:\n',Y_OOB_LIST)
```

```
y_mse values through sklearn Y_MSE_LIST_SKT:
 [2.73, 2.52, 3.09, 2.42, 3.21, 2.71, 2.81, 3.69, 2.52, 2.65, 2.64, 3.1, 2.78, 2.72, 3.32, 2.94, 2.
83, 2.55, 2.89, 2.62, 3.03, 2.71, 2.69, 2.49, 2.46, 2.82, 2.83, 2.67, 3.07, 2.84, 2.91, 2.56, 3.23,
2.65, 2.6]

MSE values calculated:
 [2.73, 2.52, 3.09, 2.42, 3.21, 2.71, 2.81, 3.69, 2.52, 2.65, 2.64, 3.1, 2.78, 2.72, 3.32, 2.94, 2.
83, 2.55, 2.89, 2.62, 3.03, 2.71, 2.69, 2.49, 2.46, 2.82, 2.83, 2.67, 3.07, 2.84, 2.91, 2.56, 3.23,
2.65, 2.6]

OOB_values:
 [172.46, 155.52, 169.05, 128.0, 95.99, 214.9, 91.75, 118.5, 191.06, 165.18, 123.23, 90.21, 104.72
, 97.95, 94.51, 119.0, 96.34, 88.51, 165.79, 105.59, 95.46, 132.46, 89.58, 93.9, 117.79, 182.43, 1
69.45, 101.75, 171.05, 133.43, 95.13, 91.79, 145.98, 88.16, 131.57]
```

In [7]:

```
#==================================================================================================
=======================#
#                                                     TASK-2
#
#==================================================================================================
=======================#
#Calculating Cinfidence Interval:
import statistics as stat
import math
y_mse_mean=stat.mean(Y_MSE_LIST)
y_oob_mean=stat.mean(Y_OOB_LIST)
y_mse_stdev=stat.stdev(Y_MSE_LIST)
y_oob_stdev=stat.stdev(Y_OOB_LIST)
print('y_mse_mean=',y_mse_mean)
print('y_oob_mean=',y_oob_mean)
print('y_mse_stdev=',y_mse_stdev)
print('y_oob_stdev=',y_oob_stdev)

n=len(Y_MSE_LIST)
CI_MSE_LOWER=(y_mse_mean-((2*y_mse_stdev)/(math.sqrt(n))))
CI_MSE_UPPER=(y_mse_mean+((2*y_mse_stdev)/(math.sqrt(n))))
CI_OOB_LOWER=(y_oob_mean-((2*y_oob_stdev)/(math.sqrt(n))))
CI_OOB_UPPER=(y_oob_mean+((2*y_oob_stdev)/(math.sqrt(n))))
#printing the values
print('CI_MSE_LOWER=',CI_MSE_LOWER)
print('CI_MSE_UPPER=',CI_MSE_UPPER)
print('CI_OOB_LOWER=',CI_OOB_LOWER)
print('CI_OOB_UPPER=',CI_OOB_UPPER)
```

```
y_mse_mean= 2.8085714285714287
y_oob_mean= 126.51971428571429
y_mse_stdev= 0.27597040396670475
y_oob_stdev= 35.83530066073645
CI_MSE_LOWER= 2.7152764041347854
CI_MSE_UPPER= 2.901866453008072
CI_OOB_LOWER= 114.40517155655547
CI_OOB_UPPER= 138.6342570148731
```

In [19]:

```
#==================================================================================================
=======================#
#                                                     TASK-3
#
#==================================================================================================
=======================#

#Calculate price of house for given X point:

xq= [0.18,20.0,5.00,0.0,0.421,5.60,72.2,7.95,7.0,30.0,19.1,372.13,18.60]
xq=np.array(xq)
xq=xq.reshape(1,-1)
xq
y_pdt=[]
for i in range(30):
    y_total_pred=[]
    clf.fit(x[:,col_set[i]][X_train[i]], y[X_train[i]])
    y_pdt.append(clf.predict(xq[:,col_set[i]]))
#computing Average y_ptd value
y_sum=0
for j in range(len(y_pdt)):
    y_sum=y_sum+y_pdt[j]
y_pdt_avg=y_sum/30
y_pdt_avg=round(float(y_pdt_avg),2)
print('Predicted Home price for the given datapoint:',y_pdt_avg)
```

```
Predicted Home price for the given datapoint: 20.3
```