Importing The Libraries

In [50]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error
import random
from sklearn.tree import DecisionTreeRegressor
from operator import add
from scipy.sparse import hstack
```

Loading The Dataset

In [51]:
```
boston = load_boston()
x=boston.data #independent variables
y=boston.target #target variable
x=pd.DataFrame(x)
```

In [52]:
```
x.head(2)
```

Out[52]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.9	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.9	9.14

Task -1

In [54]:
```
def bagging(x,y):
    index=[] #used for storing indices of points used for fitting
    prediction_model=[] # used for storing prediction of each model
    oob_pred=[]
    pred=[0]* 506
    for i in range(0,30):
        k=np.random.choice(506,size=303,replace=False) # generating 60% of points randomly
        p=np.random.choice(k,size=203,replace=True) #  generating 40% of points randomly
        rs=np.hstack((k,p))                          # from already generated 60% points
        index.append(rs)
        data=x.iloc[rs]
        col=random.randint(3,13) #randomy selecting the columns based on randint generator
        x_data=data.sample(col, axis=1)
        y_data=y[rs]
        model=DecisionTreeRegressor()
        model.fit(x_data,y_data) # fitting the model
        p=model.predict(x[x_data.columns])
        prediction_model.append(p)
        pred=list( map(add, pred, p) )
    prediction= [x / 30 for x in pred]
    mse=mean_squared_error(y,prediction) # calculating mean score error

    for j in range(506):
        count=0
        value=0
        for k in range(30):
            if j not in index[k]: # checking whether the point  it is used for fitting
                value=value+prediction_model[k][j] # if not it is used for oob score calculation
                count=count+1
        oob_pred.append(value/count)
    oob=mean_squared_error(y,oob_pred) # oob score calculation
    return mse,oob
```

In [55]:
```
mse,oob=bagging(x,y)
print("MEAN SCORE ERROR: ",mse)
print("OOB SCORE ERROR: ",oob)
```

```
MEAN SCORE ERROR:  2.151415053938947
OOB SCORE ERROR:  12.619826128479742
```

Task - 2

In [56]:
```
mse_scores=[]
oob_scores=[]
for k in range(35):
    mse,oob=bagging(x,y)
    mse_scores.append(mse)
    oob_scores.append(oob)
```

In [57]:

```
mse=np.array(mse_scores)
oob=np.array(oob_scores)
```

In [58]:

```
def ci(data):
    " calculating the confidence interval "
    mean=data.mean()
    std=data.std()
    size=len(data)
    left_limit  = np.round(mean - 2*(std/np.sqrt(size)), 3)
    right_limit = np.round(mean + 2*(std/np.sqrt(size)), 3)
    return left_limit,right_limit
```

In [59]:

```
left,right=ci(mse)
print("Confidence Interval Of MSE :[{} ,{}]" .format(left,right))
```

```
Confidence Interval Of MSE :[2.267 ,2.479]
```

In [60]:

```
left,right=ci(oob)
print("Confidence Interval Of OOB :[{} ,{}]" .format(left,right))
```

```
Confidence Interval Of OOB :[13.343 ,14.151]
```

Task -3

In [61]:

```
def predict(xq):
    final=0
    for i in range(0,30):
        query=[]
        k=np.random.choice(506,size=303,replace=False)
        p=np.random.choice(k,size=203,replace=True)
        rs=np.hstack((k,p))
        data=x.iloc[rs]
        col=random.randint(3,13)
        x_data=data.sample(col, axis=1)
        y_data=y[rs]
        model=DecisionTreeRegressor()
        model.fit(x_data,y_data)
        slic= x_data.columns
        query=[xq[i] for i in slic]
        query=np.array(query).reshape(1,-1)
        p=model.predict(query)
        final=final+p
    final=final/30
    return (final[0])
```

In [62]:

```
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]
```

In [63]:

```
predict(xq)
```

Out[63]:

```
21.23888888888889
```