Task 1

Step - 1

- **Creating samples**
  Randomly create 30 samples from the whole boston data points
  - Creating each sample: Consider any random 303(60% of 506) data points from whole data set and then replicate any 203 points from the sampled points

    For better understanding of this procedure lets check this examples, assume we have 10 data points [1,2,3,4,5,6,7,8,9,10], first we take 6 data points randomly , consider we have selected [4, 5, 7, 8, 9, 3] now we will replicate 4 points from [4, 5, 7, 8, 9, 3], consder they are [5, 8, 3,7] so our final sample will be [4, 5, 7, 8, 9, 3, 5, 8, 3,7]
- **Create 30 samples**
  - Note that as a part of the Bagging when you are taking the random samples **make sure each of the sample will have different set of columns**
    Ex: Assume we have 10 columns[1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10] for the first sample we will select [3, 4, 5, 9, 1, 2] and for the second sample [7, 9, 1, 4, 5, 6, 2] and so on... Make sure each sample will have atleast 3 feautres/columns/attributes
- **Note - While selecting the random 60% datapoints from the whole data, make sure that the selected datapoints are all exclusive, repetition is not allowed.**

Step - 2

Building High Variance Models on each of the sample and finding train MSE value

- Build a regression trees on each of 30 samples.
- Computed the predicted values of each data point(506 data points) in your corpus.
- Predicted house price of i^{th} data point y^i_{pred}
  $$= \frac{1}{30} \sum_{k=1}^{30} (\text{predicted value of } x^i \text{ with } k^{th} \text{ model})$$
- Now calculate the MSE
  $$= \frac{1}{506} \sum_{i=1}^{506} (y^i - y^i_{pred})^2$$

Step - 3

- **Calculating the OOB score**

- Predicted house price of i^{th} data point
  y^i_{pred} .
  $$= \frac{1}{k} \sum_{\text{k= model which was buit on samples not included } x^i} (\text{predicted value of } x^i \text{ with } k^{th} \text{ model})$$
- Now calculate the $OOBScore = \frac{1}{506} \sum_{i=1}^{506} (y^i - y^i_{pred})^2$.

Task 2

- **Computing CI of OOB Score and Train MSE**
  - **Repeat Task 1 for 35 times, and for each iteration store the Train MSE and OOB score **
  - **After this we will have 35 Train MSE values and 35 OOB scores**
  - **using these 35 values (assume like a sample) find the confidence intravels of MSE and OOB Score**
  - **you need to report CI of MSE and CI of OOB Score**
  - **Note: Refer the Central_Limit_theorem.ipynb to check how to find the confidence intravel**
    **
**

Task 3

- **Given a single query point predict the price of house.**

Consider 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] Predict the house price for this point as mentioned in the step 2 of Task 1.

A few key points

- **Remember that the datapoints used for calculating MSE score contain some datapoints that were initially used while training the base learners (the 60% sampling). This makes these datapoints partially seen (i.e. the datapoints used for calculating the MSE score are a mixture of seen and unseen data). Whereas, the datapoints used for calculating OOB score have only the unseen data. This makes these datapoints completely unseen and therefore appropriate for testing the model's performance on unseen data.**
- **Given the information above, if your logic is correct, the calculated MSE score should be less than the OOB score.**
- **The MSE score must lie between 0 and 10.**
- **The OOB score must lie between 10 and 35.**
- **The difference between the left nad right confidence-interval values must not be more than 10. Make sure this is true for both MSE and OOB confidence-interval values.**

Importing Libraries

In [ ]:

```
import numpy as np # importing numpy for numerical computation
from sklearn.datasets import load_boston # here we are using sklearn's boston dataset
from sklearn.metrics import mean_squared_error # importing mean_squared_error metric
from sklearn.tree import DecisionTreeRegressor
```

Loading Data

In [ ]:

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

In [ ]:

```
x.shape
```

Out[ ]:

```
(506, 13)
```

In [ ]:

```
x[:5]
```

Out[ ]:

```
array([[6.3200e-03, 1.8000e+01, 2.3100e+00, 0.0000e+00, 5.3800e-01,
        6.5750e+00, 6.5200e+01, 4.0900e+00, 1.0000e+00, 2.9600e+02,
        1.5300e+01, 3.9690e+02, 4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, 0.0000e+00, 4.6900e-01,
        6.4210e+00, 7.8900e+01, 4.9671e+00, 2.0000e+00, 2.4200e+02,
        1.7800e+01, 3.9690e+02, 9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, 0.0000e+00, 4.6900e-01,
        7.1850e+00, 6.1100e+01, 4.9671e+00, 2.0000e+00, 2.4200e+02,
        1.7800e+01, 3.9283e+02, 4.0300e+00],
       [3.2370e-02, 0.0000e+00, 2.1800e+00, 0.0000e+00, 4.5800e-01,
        6.9980e+00, 4.5800e+01, 6.0622e+00, 3.0000e+00, 2.2200e+02,
        1.8700e+01, 3.9463e+02, 2.9400e+00],
       [6.9050e-02, 0.0000e+00, 2.1800e+00, 0.0000e+00, 4.5800e-01,
        7.1470e+00, 5.4200e+01, 6.0622e+00, 3.0000e+00, 2.2200e+02,
        1.8700e+01, 3.9690e+02, 5.3300e+00]])
```

Task 1

step 1: Generating samples

In [ ]:

```
def generating_samples(input_data,target_data):
  # Getting 303 random row indices from the input data
  getting_rows=np.random.choice(list(range(len(x))),303,replace=False)
  # Extracting 203 random rows indices
  change_rows=np.random.choice(list(range(len(getting_rows))),203,replace=False)
  # Getting from 3 to 13 random column indices
  getting_columns=np.random.randint(3,13,np.random.randint(3,13))
  sample_data=(input_data[getting_rows[:,None],getting_columns])
  target_of_sample_data=(target_data[getting_rows])
  # Replicating Data
  replicated_sample_data=list(sample_data[change_rows])
  target_of_replicated_sample_data=(target_of_sample_data[change_rows])
  # performing Vstack to get full sample and target data
  final_sample_data=np.vstack([sample_data,replicated_sample_data])
  final_target_data=np.vstack([target_of_sample_data.reshape(-1,1),target_of_replicated_
sample_data.reshape(-1,1)])
  return list(final_sample_data),list(final_target_data),list(getting_rows),list(getting
_columns)
```

In [ ]:

```
# Use generating_samples function to create 30 samples
# store these created samples in a list
list_input_data =[]
list_output_data =[]
list_selected_row= []
list_selected_columns=[]
```

In [ ]:

```
for i in range(0,30):
  sample,target,oob,fi = generating_samples(x,y)
  list_input_data.append(sample)
  list_output_data.append(target)
  list_selected_row.append(oob)
  list_selected_columns.append(fi)
```

In [ ]:

```
def grader_30(a):
    assert(len(a)==30 and len(a[0])==506)
```

```
    return True
grader_30(list_input_data)
```

Out[ ]:

True

In [ ]:

```
def grader_samples(a,b,c,d):
    length = (len(a)==506  and len(b)==506)
    sampled = (len(a)-len(set([str(i) for i in a]))==203)
    rows_length = (len(c)==303)
    column_length= (len(d)>=3)
    assert(length and sampled and rows_length and column_length)
    return True
a,b,c,d = generating_samples(x, y)
grader_samples(a,b,c,d)
```

Out[ ]:

True

step 2: Building High Variance Models on each of the sample and finding train MSE value

In [ ]:

```
from sklearn.tree import DecisionTreeRegressor
all_models=[]
for i in range(0,30):
  Reg=DecisionTreeRegressor()
  Reg.fit(list_input_data[i],list_output_data[i])
  all_models.append(Reg)
```

step 3: Calculating MSE and OOB Score

In [ ]:

```
pred_values=[]
for i in range(0,30):
  pred_values.append(all_models[i].predict(x[:,list_selected_columns[i]]))
predict=np.median(np.array(pred_values),axis=0)
#mean squared error
mean_squared_error(y,predict)
```

Out[ ]:

0.16176883351340998

In [ ]:

```
oob_score=[]
for i in range(len(x)):
    predict_oob=[]
    for j in range(0,30):
        if i not in list_selected_row[j]:
            predict_oob.append(all_models[j].predict(x[i][list_selected_columns[j]].resh
ape(1,-1)))
    oob_score.append(np.median(predict_oob))
print("oob_score is : ",(mean_squared_error(y,oob_score)))
```

```
oob_score is :  15.586505467279439
```

Task 2

Computing CI of OOB Score and Train MSE

In [ ]:

```
from sklearn.tree import DecisionTreeRegressor
from tqdm.notebook import tqdm

obb_score_full=[]
mse_full=[]
for i in tqdm(range(35)):        # Repeating Task 1 for 35 times
  input_data =[]
  output_data =[]
  selected_row= []
  selected_columns=[]
  for i in range(0,30):
    sample,target,oob,fi = generating_samples(x,y)    # Calling generate_samples function
    input_data.append(sample)
    output_data.append(target)
    selected_row.append(oob)
    selected_columns.append(fi)

  pred_values=[]
  all_models=[]
  for i in range(0,30):
    model=DecisionTreeRegressor()               # Calling Regressor Model
    model.fit(input_data[i],output_data[i])
    all_models.append(model)
    pred_values.append(all_models[i].predict(x[:,selected_columns[i]]))
  predict=np.median(np.array(pred_values),axis=0)
  #mean squared error
  mse_full.append(mean_squared_error(y,predict))    # appending 35 Train MSE values
  obb_predict=[]
  oob_score=[]
  for i in range(len(x)):
      predict_oob=[]
      for j in range(0,30):
          if i not in selected_row[j]:
              predict_oob.append(all_models[j].predict(x[i][selected_columns[j]].reshape
(1,-1)))
      oob_score.append(np.median(predict_oob))
  obb_score_full.append(mean_squared_error(y,oob_score))      # appending 35 OOB scores
obb_score_full=np.array(obb_score_full)
mse_full=np.array(mse_full)
```

In [ ]:

```
import math as m
def ci(data):
    " calculating the confidence interval "
    mean = data.mean()
    std = data.std()
    size = len(data)
    left_limit  = np.round(mean - 2*(std/m.sqrt(size)), 3)
    right_limit = np.round(mean + 2*(std/m.sqrt(size)), 3)
    return left_limit,right_limit
left,right=ci(mse_full)
print("Confidence Interval Of MSE :",left,'to',right)

left,right=ci(obb_score_full)
print("Confidence Interval Of OOB :",left,'to',right)
```

```
Confidence Interval Of MSE : 0.085 to 0.146
Confidence Interval Of OOB : 15.209 to 16.413
```

Task 3

Predict the house price

In [ ]:

```
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]

def predict_y_when_x(x_query):
  y_pred_array_30_sample = []

  for i in range(0, 30):
    model_i = all_models[i]   # storing all models
    # Extract x for ith data point with specific number of featues from list_selected_col
umns
    x_data_point_i = [x_query[column] for column in selected_columns[i]]
    x_data_point_i = np.array(x_data_point_i).reshape(1, -1)
    y_pred_i = model_i.predict(x_data_point_i)
    y_pred_array_30_sample.append(y_pred_i)

  y_pred_array_30_sample = np.array(y_pred_array_30_sample)
  y_pred_median = np.median(y_pred_array_30_sample)
  return y_pred_median

y_pred_for_xq = predict_y_when_x(xq)
print(y_pred_for_xq)
```

```
18.5
```

Observations from task 1,2 & 3

In [ ]:

```
from prettytable import PrettyTable  # http://zetcode.com/python/prettytable/
from prettytable import ALL as ALL
table=PrettyTable(hrules=ALL)
#table.field_names = [ "Task","", "", "Hyper Parameter", "Train-AUC","Test-AUC"] #
table.add_row([1, "Mean Square Error",0.16176])
table.add_row([2, "OOB Score", 15.58650])
table.add_row([3,"Confidence Interval Of MSE ","0.085 - 0.146"])
table.add_row([4, " Confidence Interval Of OOB","15.209 - 16.413"])
table.add_row([5,"Predicted Value", 18.5])
print(table)
```

```
+---------+-----------------------------+------------------+
| Field 1 |           Field 2           |     Field 3      |
+---------+-----------------------------+------------------+
|    1    |      Mean Square Error      |     0.16176      |
+---------+-----------------------------+------------------+
|    2    |          OOB Score          |     15.5865      |
+---------+-----------------------------+------------------+
|    3    | Confidence Interval Of MSE  |  0.085 - 0.146   |
+---------+-----------------------------+------------------+
|    4    |  Confidence Interval Of OOB | 15.209 - 16.413  |
+---------+-----------------------------+------------------+
|    5    |       Predicted Value       |       18.5       |
+---------+-----------------------------+------------------+
```