Bootstrap assignment

There will be some functions that start with the word "grader" ex: grader_sampples(), grader_30().. etc, you should not change those function definition.

Every Grader function has to return True.

Importing packages

```
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
import random
```

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

```
x
```

```
array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
        4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
        9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
        4.0300e+00],
       ...,
       [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        5.6400e+00],
       [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
        6.4800e+00],
       [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        7.8800e+00]])
```

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

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
  $y^i_{pred} = \frac{1}{k}\sum_{\text{k= model which was buit on samples not included } x^i}(\text{predicted value of } x^i \text{ with } k^{th}$]
  .
- 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.

Task - 1

Step - 1

- **Creating samples**

Algorithm

Pesudo Code for generating Sample

```
def generating_samples(input_data, target_data):

    Selecting_rows <--- Getting 303 random row indices from the input_data

    Replcaing_rows <--- Extracting 206 random row indices from the "Selecting_rows"

   Selecting_columns<--- Getting from 3 to 13 random column indices

   sample_data<--- input_data[Selecting_rows[:,None],Selecting_columns]

   target_of_sample_data <--- target_data[Selecting_rows]

    #Replicating Data

   Replicated_sample_data <--- sample_data [Replaceing_rows]

   target_of_Replicated_sample_data<--- target_data[Replaceing_rows]

    # Concatinating data

    final_sample_data <---  perform vertical stack on  sample_data, Replicated_sample_data

    final_target_data<--- perform vertical stack on target_of_sample_data.reshape(-1,1), target_of_Replicated_sample_data.reshape(-1,1)

    return final_sample_data,  final_target_data, Selecting_rows, Selecting_columns
```

```
def generating_samples(input_data, target_data):
      selecting_rows_1 = random.sample(range(0,506),303)
      selecting_rows = np.array(selecting_rows_1)
      replicating_rows_1 = random.sample(selecting_rows_1,203)
      replicating_rows = np.array(replicating_rows_1)
      num_selecting_columns = random.randint(3,13)
      selecting_columns = np.array(random.sample(range(0,13),num_selecting_columns))
      sample_data = input_data[selecting_rows[:,None],selecting_columns]
      target_of_sample_data = target_data[selecting_rows_1]

      replicated_sample_data = input_data[replicating_rows[:,None], selecting_columns]
      target_of_replicated_data = target_data[replicating_rows]

      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
      return final_sample_data, final_target_data, selecting_rows, selecting_columns
```

- **Write code for generating samples**

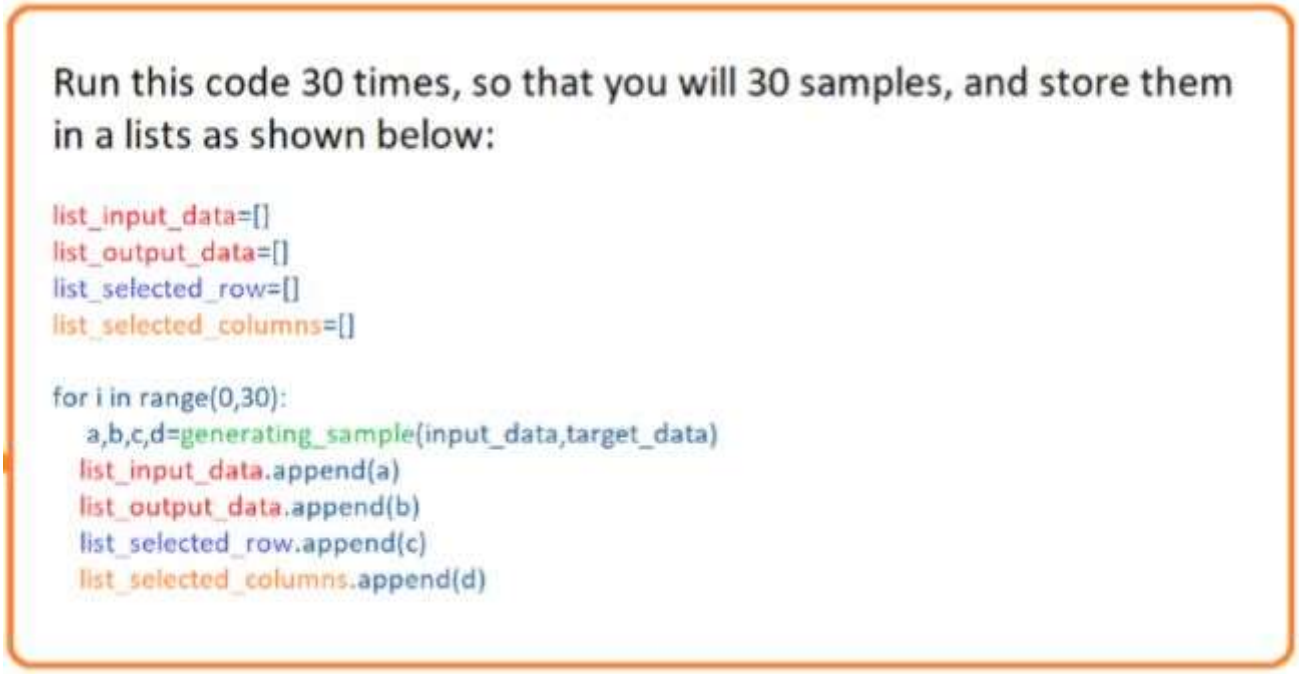
Grader function - 1

```
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)
```

```
True
```

- **Create 30 samples**




```
# 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=[]

for i in range(0,30):
  a,b,c,d, = generating_samples(x, y)
  list_input_data.append(a)
  list_output_data.append(b)
  list_selected_row.append(c)
  list_selected_columns.append(d)
```

Grader function - 2

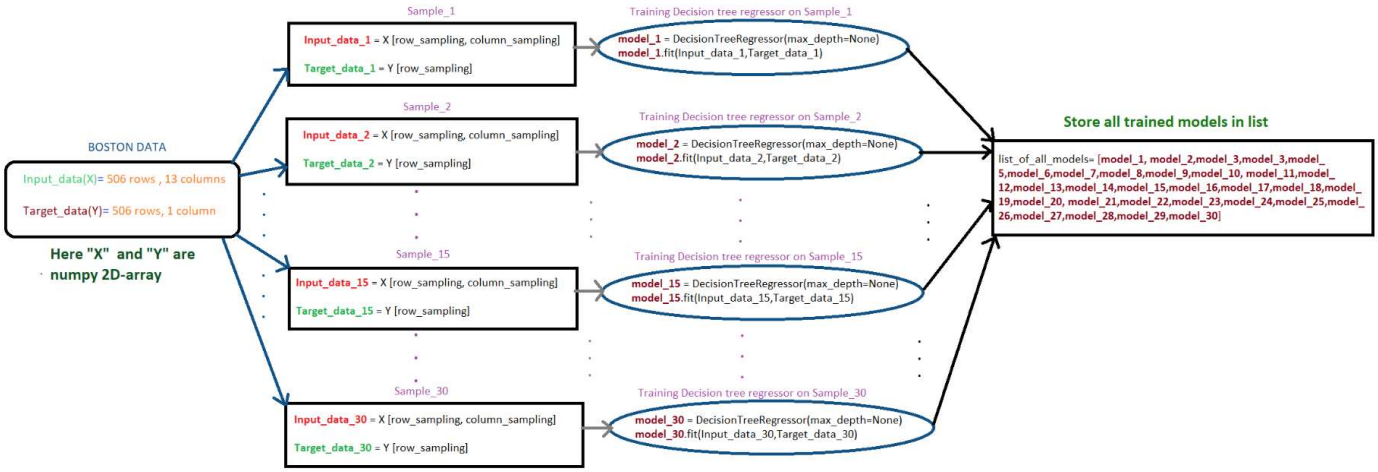
```
def grader_30(a):
    assert(len(a)==30 and len(a[0])==506)
    return True
```

```
grader_30(list_input_data)
```

```
True
```

Step - 2

Flowchart for building tree

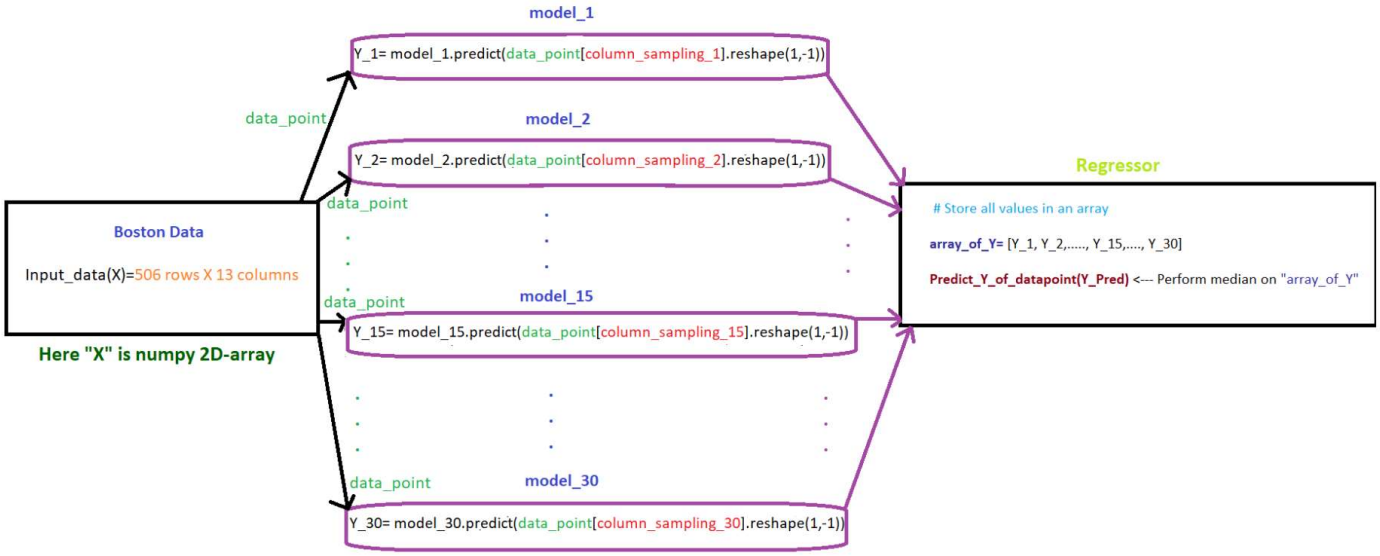


- **Write code for building regression trees**

```
from sklearn.tree import DecisionTreeRegressor
list_of_all_models = []

for input_data, taget_data in zip(list_input_data, list_output_data):
  regressor = DecisionTreeRegressor(max_depth=None)
  regressor.fit(input_data,taget_data)
  list_of_all_models.append(regressor)


regressor1 = DecisionTreeRegressor(max_depth=None)
regressor1.fit(list_input_data[1],list_output_data[1])
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

Flowchart for calculating MSE



After getting predicted_y for each data point, we can use sklearns mean_squared_error to calculate the MSE between predicted_y and actual_y.

- **Write code for calculating MSE**

```
array_of_y = []
for i in range(0,30):
  data_point = x[:, list_selected_columns[i]]
  y = list_of_all_models[i].predict(data_point)
  array_of_y.append(y)
```

```
predict_y_of_datapoint = np.array(array_of_y)
predict_y_of_datapoint=predict_y_of_datapoint.transpose()
mean_predict_y_of_datapoint = np.mean(predict_y_of_datapoint, axis=1)
mean_predict_y_of_datapoint.shape
```

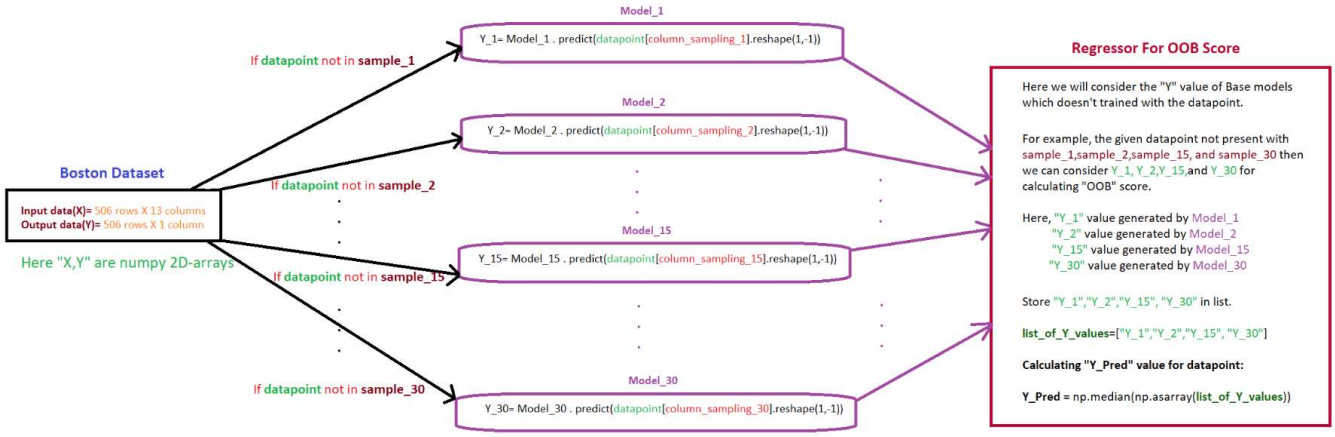
```
(506,)
```

```
from sklearn.metrics import mean_squared_error
from statistics import median
print(mean_squared_error(y, mean_predict_y_of_datapoint))
```

```
7.580992497573695
```

Step - 3

Flowchart for calculating OOB score



Now calculate the $OOBScore = \frac{1}{506}\sum_{i=1}^{506}(y^i - y^i_{pred})^2$.

- **Write code for calculating OOB score**

```
predicted_y = []
for i, point in enumerate(x):
  model_y_pred = []
  for j, model in enumerate(list_of_all_models):
    if i not in list_selected_row[j]:
      model_y_pred.append(model.predict(point[list_selected_columns[j]].reshape(1,-1)))
  predicted_y.append(np.median(np.asarray(model_y_pred)))
OOB_Score = mean_squared_error(y,np.asarray(predicted_y))


print("OOB Score is: ",OOB_Score)
```

```
    OOB Score is:  16.434470855902013
```

Task 2

```
def task2(x,y):
  list_input_data =[]
```

```
    list_output_data =[]
    list_selected_row= []
    list_selected_columns=[]

    for i in range(0,30):
      a,b,c,d, = generating_samples(x, y)
      list_input_data.append(a)
      list_output_data.append(b)
      list_selected_row.append(c)
      list_selected_columns.append(d)

    list_of_all_models = []
    for input_data, taget_data in zip(list_input_data, list_output_data):
      regressor = DecisionTreeRegressor(max_depth=None)
      regressor.fit(input_data,taget_data)
      list_of_all_models.append(regressor)



    # calculating MSE
    predict_y_of_datapoint = np.array(array_of_y)
    predict_y_of_datapoint=predict_y_of_datapoint.transpose()
    mean_predict_y_of_datapoint = np.mean(predict_y_of_datapoint, axis=1)
    mean_predict_y_of_datapoint.shape
    MSE = mean_squared_error(y, mean_predict_y_of_datapoint)

    # calculating oob score
    predicted_y = []
    for i, point in enumerate(x):
      model_y_pred = []
      for j, model in enumerate(list_of_all_models):
        if i not in list_selected_row[j]:
          model_y_pred.append(model.predict(point[list_selected_columns[j]].reshape(1,-1)))
      predicted_y.append(np.median(np.asarray(model_y_pred)))
    OOB_Score = mean_squared_error(y,np.asarray(predicted_y))



    return MSE, OOB_Score



  from sklearn.tree import DecisionTreeRegressor
  from sklearn.metrics import mean_squared_error
  boston = load_boston()
  x=boston.data #independent variables
  y=boston.target #target variable
  MSE = []
  OOB_SCORE = []
  for i in range(0,35):
    mse, obb = task2(x,y)
    MSE.append(mse)
    OOB_SCORE.append(obb)
```

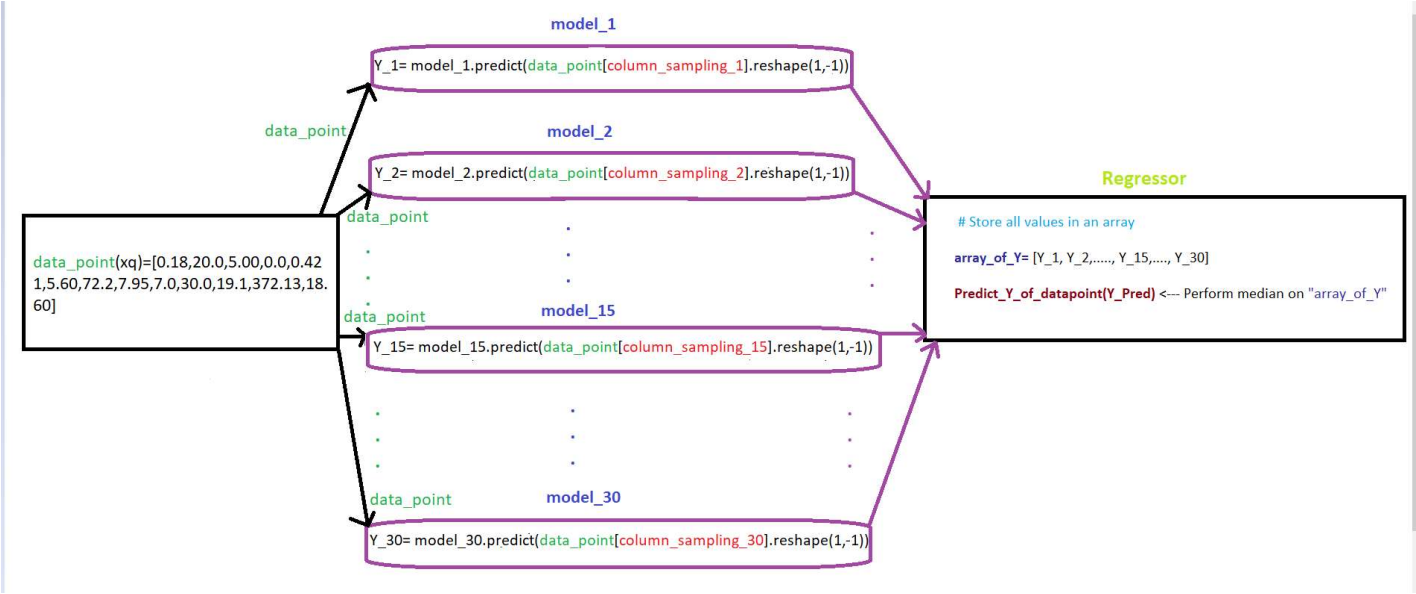
```
import scipy
confidence_level = 0.95

degrees_freedom = 34
sample_mean = np.mean(MSE)

sample_standard_error = scipy.stats.sem(MSE)


confidence_interval = scipy.stats.t.interval(confidence_level, degrees_freedom, sample_mean,


sample_mean = np.mean(OOB_SCORE)

sample_standard_error = scipy.stats.sem(OOB_SCORE)


confidence_interval_ob = scipy.stats.t.interval(confidence_level, degrees_freedom, sample_mea


print("CI MSE: ", confidence_interval)
print("CI OOB: ", confidence_interval_ob )
```

```
CI MSE:  (2.3308125824807377, 2.3308125824807377)
CI OOB:  (13.384057246446394, 14.581856304453103)
```

Task 3

Flowchart for Task 3

Hint: We created 30 models by using 30 samples in TASK-1. Here, we need send query point "xq" to 30 models and perform the regression on the output generated by 30 models.



- **Write code for TASK 3**

```
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]
predicted_y_array = []
for i in range(0,30):

  model = list_of_all_models[i]
  x_data = [xq[col] for col in list_selected_columns[i]]
  x_data = np.array(x_data).reshape(1, -1)
  y = model.predict(x_data)
  predicted_y_array.append(y)


y = np.array(predicted_y_array)
predicted_y = np.median(y)
print(predicted_y)
```

```
18.5
```

Write observations for task 1, task 2, task 3 indetail

Task 1

For Task 1 we went through three steps:

Step 1: We created samples that are randomly generated from the boston datasets. For creating samples first we are going to create the 60% of the random samples of 506 data points and remaining 40% data points will be replicated. We need to take different set of columns for each of the samples, and each samples will have atleast 3 features, columns or attributes.

Step 2: Now in step 2 we are going to create high variance model on each of the sample for that we are going to create a regression tree with high variance for each of the 30 samples. Now we are going to get the prediction values for each 506 datapoints. Afterwards we will calculate MSE i.e Mean Squared Error value. For calculating MSE we will first get labels by predicting for each model on their specific columns and take the median on predictions.

Step 3: In this step we are going to calculate the OOB (Out of Bag) score for all 506 for that in oob instead of predicting for all points for each model we will check whether that point is present in the training set of the model, if it is not present then only you will predict using that model.

Task 2

In Task 2 we are going to calculate the Confidence Interval of OOB Score and Train MSE. For that we need to repeat the Task 1 for 35 times and we need to store the Train MSE and OOB score for each iterations. After getting 35 Train MSE and 35 OOB score, using these value we are going to find the Confidence Interval of Train MSE and OOB Score.

Task 3

In Task 3 we need consider a single query point i.e 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] for Predicting the house price for this point as mentioned in the step 2 of Task 1.

✓ 0s completed at 21:24