Bootstrap assignment

There will be some functions that start with the word "grader" ex: grader_samples(), grader_30() , etc, you should not change those function definition.
Every Grader function has to return True.

Importing packages

```
In [158]: 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
          from statistics import median
```

```
In [159]: boston = load_boston()
          x=boston.data #independent variables
          y=boston.target #target variable
```

```
In [164]: len(x)
Out[164]: 506
```

```
In [161]: y.shape
Out[161]: (506,)
```

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}, consider 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/fetures
- **Note - While selecting the random 60% datapoints from the whole data, make sure that the selected**

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 points(506 data points) in your corpus.
- Predicted house price of i^{th} data point $y_{pred}^i = \frac{1}{30}\sum_{k=1}^{30}$ (predicted value of x^i with k^{th} model)
- Now calculate the $MSE = \frac{1}{506}\sum_{i=1}^{506}(y^i - y_{pred}^i)^2$

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**
    <hul>

Task 3

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

Consider xq= [0.10,20.0,5.00,0.0,421.5,60,72.2,7.95,7.0,30.0,19.1,372.13,18.69] 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 (for 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.**

Task - 1

Step - 1

- **Creating samples**

Algorithm

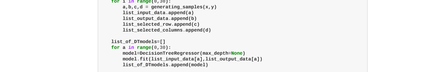
- **Write code for generating samples**

```
In [251]: def generating_samples(input_data, target_data):


              selecting_rows = np.random.choice(len(input_data), 303, replace=False)
              ### https://www.geeksforgeeks.org/numpy-random-choice-in-python/
              replacing_rows = np.random.choice(selecting_rows, 203, replace=False)

              ### https://www.geeksforgeeks.org/python-random-sample-function/
              ### https://www.w3schools.com/python/ref_random_randint.asp
              selecting_columns = np.array(random.sample(range(0, 13), random.randint(3, 13) ))
              sample_data = input_data[selecting_rows[:, None], selecting_columns]
              target_of_sample_data = target_data[selecting_rows]


              replicated_sample_data = input_data[replacing_rows[:, None], selecting_columns ]
              target_replicated_sample_data = target_data[replacing_rows]


              final_sample_data = np.vstack((sample_data, replicated_sample_data ))
              final_target_data = np.vstack((target_of_sample_data.reshape(-1, 1), target_replicated_s
          ample_data.reshape(-1, 1)))

              return final_sample_data, final_target_data, selecting_rows, selecting_columns
```

Grader function - 1

```
In [202]: 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[202]: True
```

- **Create 30 samples**



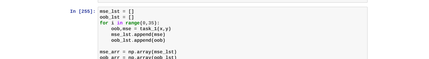

```
In [203]: # Use generating_samples function to create 30 samples
          # store these created samples in a list
          list_input_data =[]
          list_output_data =[]
          list_row_values =[]
          list_selected_columns=[]
          for i in range(0,30):
              a,b,c,d = 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

```
In [204]: def grader_30(a):
              assert(len(a)==30 and len(a[0])==506)
              return True
          grader_30(list_input_data)
Out[204]: True
```

Step - 2

Flowchart for building tree



- **Write code for building regression trees**

```
In [205]: ## https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/

          from sklearn.tree import DecisionTreeRegressor
          list_of_DTmodels=[]
          for a in range(0,30):
              model=DecisionTreeRegressor(max_depth=None)
              model.fit(list_input_data[a],list_output_data[a])
              list_of_DTmodels.append(model)
```

Flowchart for calculating MSE

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

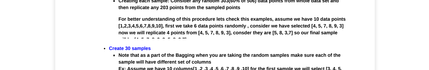
- **Write code for calculating MSE**

```
In [238]: new_array= []

          for i in range(0, 30):
              point = x[:,list_selected_columns[i]]
              y_cap = list_of_DTmodels[i].predict(point)
              new_array.append(y_cap)

          y_hat = np.array(new_array)
          y_hat = y_hat.transpose()
          ### https://datagy.io/mean-squared-error-python/
          y_pred= np.median(y_hat, axis=1)
          y_pred.shape
          print("MSE : ", mean_squared_error(y, y_pred ))
```

```
MSE :  0.042699167804187973
```

Step - 3

Flowchart for calculating OOB score



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

- **Write code for calculating OOB score**

```
In [245]: ### https://corbanosli.github.io/posts/random-forests-from-scratch/

          y_median_list = []

          for i in range(0, 506):
              indexs = []
              for val in range(0, 30):
                  if i not in list_selected_row[val]:
                      indexs.append(val)
              y_prebodh_list = []
              for b in indexs:
                  x_oob = x[:,list_of_DTmodels[b]
                  x_oob = [x[i][col] for col in list_selected_columns[b]]
                  x_oob = np.array(x_oob).reshape(1, -1)
                  y_predict = oob_mod.predict(x_oob)
                  y_prebodh_list.append(y_predict)

              med = np.median(np.array(y_prebodh_list))
              y_median_list.append(med)

          rows_len=506
          oob_s = 0

          for i in range(0,506):
              Diff = y[i] - y_median_list[i]
              oob_s = oob_s + ((Diff ) ** 2)
              final_oob_score = oob_s/rows_len

          print("final_oob_score is ", final_oob_score)
```

```
final_oob_score is  10.557508891215664
```

Task 2

```
In [249]: def task_1(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_DTmodels=[]
              for a in range(0,30):
                  model=DecisionTreeRegressor(max_depth=None)
                  model.fit(list_input_data[a],list_output_data[a])
                  list_of_DTmodels.append(model)

              new_array= []

              for i in range(0, 30):
                  point = x[:,list_selected_columns[i]]
                  y_cap = list_of_DTmodels[i].predict(point)
                  new_array.append(y_cap)
              y_hat = np.array(new_array)
              y_hat = y_hat.transpose()
              y_pred= np.median(y_hat, axis=1)

              MSE = mean_squared_error(y, y_pred )

              y_median_list = []

              for j in range(0, 506):
                  indexs = []
                  for val in range(0, 30):
                      if j not in list_selected_row[val]:
                          indexs.append(val)
                  y_prebodh_list = []
                  for n in indexs:
                      oob_mod = list_of_DTmodels[n]
                      row = x[j]
                      x_oob = [row[col] for col in list_selected_columns[n] ]
                      x_oob = np.array(x_oob).reshape(1, -1)
                      y_predict = oob_mod.predict(x_oob)
                      y_prebodh_list.append(y_predict)

                  med = np.median(np.array(y_prebodh_list))
                  y_median_list.append(med)

              rows_len=506
              oob_s = 0
              for i in range(0,506):
                  Diff = y[i] - y_median_list[i]
                  oob_s = oob_s + ((Diff)**2)
                  final_oob= oob_s/rows_len

              return final_oob,MSE
```

```
In [255]: mse_lst = []
          oob_lst = []
          for i in range(0,35):
              oob,mse = task_1(x,y)
              mse_lst.append(mse)
              oob_lst.append(oob)

          mse_arr = np.array(mse_lst)
          oob_arr = np.array(oob_lst)

          ## Refer Central_Limit_theorem.ipynb
          def Conf_int(data):
              mean_arr=data.mean()
              std_arr=data.std()
              size=len(data)
              left_limit = np.round(mean_arr - 2*(std_arr/np.sqrt(size)), 3)
              right_limit = np.round(mean_arr + 2*(std_arr/np.sqrt(size)), 3)
              return left_limit,right_limit
```

```
In [252]: left_mse,right_mse = Conf_int(mse_arr)
          print(left_mse)
          print(right_mse)
```

```
9.989
9.159
```

```
In [253]: left_oob,right_oob = Conf_int(oob_arr)
          print(left_oob)
          print(right_oob)
```

```
13.412
14.466
```

Task 3


```
In [254]: xq= [0.10,20.0,5.00,0.0,421.5,60,72.2,7.95,7.0,30.0,19.1,372.13,18.69]

          y_list=[]
          for i in range(0,30):
              oob_mod = list_of_DTmodels[i]
              x_oob = [xq[col] for col in list_selected_columns[i]]
              x_oob = np.array(x_oob).reshape(1, -1)
              y_predict = oob_mod.predict(x_oob)
              y_list.append(y_predict)
          y_arr=np.array(y_list)
          medi = np.median(y_arr)
          print(medi)
```

```
10.9
```

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

Observation:

Task 1:

1. **MSE value :0.0426**
2. **OOB Score :10.5575**

Task 2:

1. **Confidence interval of MSE = (9.099,9.159)**
2. **Confidence interval of OOB = (13.412,14.466)**

Task 3:

The predicted house price = 10.9