**Gradient Boosting Regressor from scratch without using Sklearn Library:**

Ensemble models are machine learning models where we combine multiple other ML models for the prediction process.

Gradient Boosting regressor is one such type of ensemble model that combines multiple weak "learners" (usually a shallow decision tree) into a single strong learner in an iterative fashion.

It is a technique for repeatedly adding decision trees so that the next decision tree corrects the previous decision tree’s error.

The model starts with the constant such as the mean of the labels. **In subsequent stages, the decision trees or the estimators are fitted to predict the negative gradients of the samples.** The gradients are updated in each iterator (for every subsequent estimator). A learning rate is used to shrink the outcome or the contribution from each subsequent tree.

**About the Python code**

I have used a dataset named [dia.csv](http://localhost:8888/edit/Resume/Projects/Resume%20-Projects/Gradient%20Boosting%20Regressor%20from%20scratch/dia.csv). (diamond price prediction dataset from Kaggle)

The feature in the dataset is given by: **Carat, Depth, Table, x, y, z**

The label in the dataset is given by:  **price**

I have created a function that would calculate the gradient boosting regressor from scratch and calculate the mape value for it. the function also has the gradient boosting algorithm to predict the label and calculate the mape value for the same.

The function takes the following values as input:

X: Features to be given to train and predict

Y: Label to be given to train

n: number of estimators/iterations for which the decision trees would be created

tdepth: Depth of the decision trees

The same estimators which I have used for the code which runs from scratch are used in the gradient-boosting regressor algorithm.

Here, I have coded the Gradient Boosting Regressor using NumPy, Pandas libraries, and decision tree regressor from the sklearn library from scratch and made predictions for the

Dataset. I have given the mean absolute percentage error values of the Algorithm coded from scratch and the sklearn model below.

MAPE value of algorithm coded from scratch: 0.26798990855195376

MAPE value of sklearn model: 0.267989908551948

Here there is a negligible difference between the two values at 10^(-16)