# Assignment 4

## Boston Housing Dataset

### Predicting Median value of owner-occupied homes

The aim of this assignment is to learn the application of machine learning algorithms to data sets. This involves learning what data means, how to handle data, training, cross validation, prediction, testing your model, etc.

This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. It was obtained from the StatLib archive, and has been used extensively throughout the literature to benchmark algorithms. The data was originally published by Harrison, D. and Rubinfeld, D.L. *Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978.*

The dataset is small in size with only 506 cases. It can be used to predict the median value of a home, which is done here. There are 14 attributes in each case of the dataset. They are:

1. `CRIM` - per capita crime rate by town
2. `ZN` - proportion of residential land zoned for lots over 25,000 sq.ft.
3. `INDUS` - proportion of non-retail business acres per town.
4. `CHAS` - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. `NOX` - nitric oxides concentration (parts per 10 million)
6. `RM` - average number of rooms per dwelling
7. `AGE` - proportion of owner-occupied units built prior to 1940
8. `DIS` - weighted distances to five Boston employment centres
9. `RAD` - index of accessibility to radial highways
10. `TAX` - full-value property-tax rate per $10,000
11. `PTRATIO` - pupil-teacher ratio by town
12. `B` - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. `LSTAT` - % lower status of the population
14. `MEDV` - Median value of owner-occupied homes in $1000's

## Aim

- To implement a linear regression with regularization via gradient descent.
- to implement gradient descent with Lp norm, for 3 different values of p in (1,2]
- To contrast the difference between performance of linear regression Lp norm and L2 norm for these 3 different values.

- Tally that the gradient descent for L2 gives same result as matrix inversion based solution.

All the code is written in a single python file. The python program accepts the data directory path as input where the train and test csv files reside. Note that the data directory will contain two files `train.csv` used to train your model and `test.csv` for which the output predictions are to be made. The output predictions get written to a file named `output.csv`. The `output.csv` file should have two comma separated columns [ID,Output].

# Working of Code

- `NumPy` library would be required, so code begins by importing it
- Import `phi` and `phi_test` from train and test datasets using `NumPy`'s `loadtxt` function
- Import `y` from train dataset using the `loadtxt` function
- Concatenate coloumn of 1s to right of `phi` and `phi_test`
- Apply min max scaling on each coloumn of `phi` and `phi_test`
- Apply log scaling on `y`
- Define a function to calculate change in error function based on `phi`, `w` and `p` norm
- Make a dictionary containing filenames as keys and `p` as values
- For each item in this dictionary
    - Set the `w` to all 0s
    - Set an appropriate value for `lambda` and `step size`
    - Calculate new value of `w`
    - Repeat steps until error between consecutive `w`s is less than threshold
    - Load values of `id` from test data file
    - Calculate `y` for test data using `phi test` and applying inverse log
    - Save the `id`s and `y` according to filename from dictionary

# Feature Engineering

- Columns of `phi` are not in same range, this is because their units are different i.e `phi` is ill conditioned
- So, min max scaling for each column is applied to bring them in range 0-1
- Same scaling would be required on columns of `phi test`
- Log scaling was used on `y`. This was determined by trial and error

# Comparison of performance

**(`p1`=1.75, `p2`=1.5, `p3`=1.3)**

- As p decreases error in y decreases

- As p decreases norm of w increases but this can be taken care by increasing lambda
- As p decreases number of iterations required decreases

## Tuning of Hyperparameter

- If `p` is fixed and `lambda` is increased error decreases up to a certain `lambda`, then it starts rising
- So, `lambda` was tuned by trial and error.
- Starting with 0, `lambda` was increased in small steps until a minimum error was achieved.

## Comparison of L2 gradient descent and closed form

- Error from L2 Gradient descent were 4.43268 and that from closed form solution was 4.52624.
- Errors are comparable so, the L2 gradient descent performs closely with closed form solution.