# Applied Artificial Intelligence Project 5

Codename: Chillmonger

Project: Zillow's Home Value Prediction (Zestimate) on Kaggle

In this competition, Zillow is asking us to predict the log-error between their Zestimate and the actual sale price, given all the features of a home. The log error is defined as

$$logerror=log(Zestimate)-log(SalePrice)$$

and it is recorded in the transactions file train.csv. In this competition, we are going to predict the logerror for the months in Fall 2017.

## Environment used and Procedure:

Python, numpy, sklearn, pandas, xgboost, anaconda.

Xgboost can be installed for anaconda by following this procedure: https://www.ibm.com/developerworks/community/blogs/jfp/entry/Installing_XGBoost_For_Anaconda_on_Windows?lang=en

Note: you can comment out the first three lines of xgb.py if xgboost is already setup properly on your system.

Note: comment out the first three lines of xgb.py and change all the paths to '../input/file_name' format if you are running the code on Kaggle.

Copy the files train_2016_v2.csv, properties_2016.csv and sample_submission.csv into the same folder as xgb.py and execute xgb.py using the following command

```
python xgb.py
```

A file named xgb.csv will be generated which can be submitted to Kaggle.

## Approach:

Since the dataset is huge, I took a computationally less intensive approach to solve the problem. I used XGBoost (eXtreme Gradient Boosting) which is an advanced implementation of the traditional gradient boosting algorithm. XGBoost implements parallel processing and is very faster compared to some other methods. It also has built-in cross validation at each iteration of the boosting process and therefore easy to get the optimum number of boosting iterations in a single run.

Initially I performed some preprocessing on the training data by dropping some unnecessary features and filling in the missing values with a default value. Then I split the training data into training and validation data into a 80:20 split using sklearn cross validation.

For the XGBoost implementation I used the default tree based model to run at each iteration and used a learning rate of 0.02. I used a max depth of 4 for the tree based model to avoid overfitting. I used linear regression as the objective and used the mean absolute error as the metric for validation data. I got a training mean absolute error of 0.067 and validation mean absolute error of 0.068 approximately before it converges.

I got a private score of 0.0764905 and a public score of 0.0651216 on Kaggle.