# Capstone Project Proposal

Machine Learning Engineer Nanodegree

Celso Araujo

April 17th, 2016

## Quick summary

Use of Kaggle's competition "House Prices: Advanced Regression Techniques" data to train and fine tune a few different regression models, compare them and combine them.

## Domain background

The problem proposed here comes from a Kaggle's competition. It is stated that competitors should build models to predict house prices based on a data set of house prices from Ames, Iowa, USA (the "Ames Housing Dataset"). The dataset contains several input variables ("features") for each house, and the sale price for about half of the houses. House price is a continuous, real-valued and nonnegative variable, so this fact suggests that regression models could be tried to solve the problem.

URL for the competition: https://www.kaggle.com/c/house-prices-advanced-regression-techniques

House price prediction can be an interesting problem for a few reasons, including but not limited to:

- It's a good problem to be used to practice, study, test and compare machine learning tools such as regression algorithms, cross-validation techniques, feature engineering and scoring metrics;
- If house prices can be (somehow) accurately predicted, then a regression model could be used to help potential buyers and sellers to establish a “fair” price, or to assess whether a given house is under or over valuated when compared to similar houses.

Personally, my motivation to chose this problem as a capstone project is twofold: first, it is a well documented problem with a readily available benchmark (the competition leaderboard itself can be used); and, second, I have professionally worked on some regression models for oil industry several years ago, so it would be nice to revive some old days.

## Problem statement

We are provided with the "Ames Housing Dataset", which contains a number of houses, each house being described by a number of features (size, location, number of bedrooms etc). We know the sale price of roughly half of the houses (training set); the goal is to predict the sale prices for the other half (test set).

Data preprocessing may include handling missing values, codifying categorical variables as numeric variables, and feature engineering (variable transformations, summing/combining and discarding variables etc). Different regression methods will be used to tackle the problem, like linear regression and gradient boosting trees. Model parameters will be fine tuned with help of k-fold cross validation and grid search. In the end, the best methods will be compared against each other and a new model being a combination of the best models will be tried and tested.

## Dataset and inputs

URL for the dataset: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

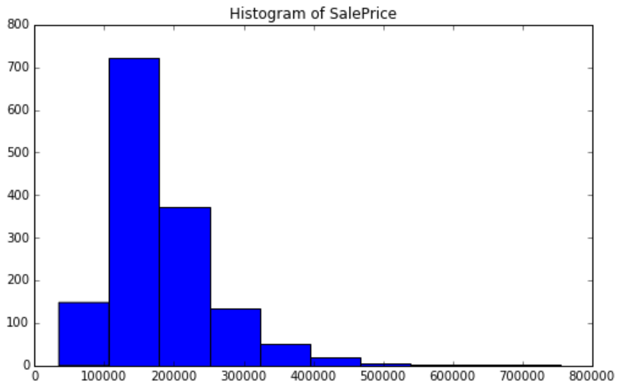
### Data size:

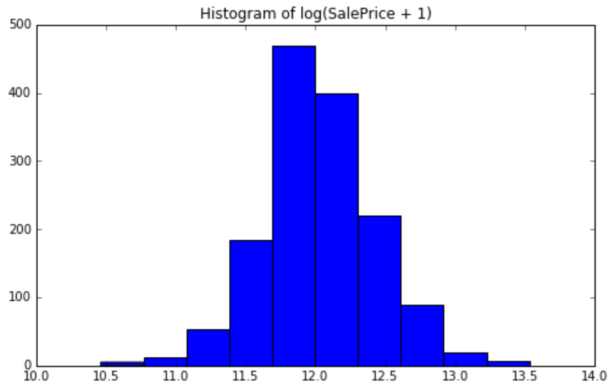
- 2919 instances (houses)
- 79 features (input variables)
- 1 target variable (sale price, real-valued)
  - 1460 houses have their sale price listed (training set)
  - 1459 houses are not provided with their sale prices (test set)

### Features:

- 26 numerical features (not counting date-related features or the target variable here). Examples: lot size, living area above ground, number of bathrooms
- 53 categorical features
  - 21 may be considered ordinal. Examples: kitchen quality, overall house condition, interior finish of garage
  - 5 date-related. Examples: year sold, year garage was built
  - 27 other categorical features. Examples: electrical system, type of foundation, neighborhood

The target variable, SalePrice, on the training set, is quite skewed. However, if we apply a log transform on it (actualy, log(SalePrice + 1), its distribution becomes more similar to a normal curve. Minimum SalePrice is *34,900*, maximum is *755,000*, the mean value is *180,921* and the median is *163,000*.





## Solution statement

The goal is to predict unknown sale prices using information from houses with known sale prices (training set). A regression model will look at the features of the training set and try to (approximately) write the sale prices in terms of the feature values. This information, this calculated relation, will be used to predict the unknown sale prices (test set), applying the calculated model to the features of the test set.

A very simple regression model (not necessarily applicable to this case!) can be like this:

$$Y = a * X + b$$

where *Y* is sale price, *X* is house size and *a* and *b* are terms calculated by the model.

## Evaluation metrics

Kaggle's competition uses the RMSLE (root mean squared logarithmic error) as metric. I will use the same metric in the project. What is RMSLE? It is a variation of the commonly-used RMSE measure for regression error. If *N* is the sample size, $p_i$ is the predicted value for the *i*-th item and $a_i$ is its the actual value, then RMSE is:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{N} (p_i - a_i)^2}$$

The closer RMSE is to zero, the better our predictions are: *RMSE = 0* means all predictions are correct.

RMSLE is the RMSE calculated over the logarithm of the predicted and actual values (in fact, we will use *log(value + 1)* to avoid calculating the logarithm of zero). If *log(x)* is the natural logarithm of *x*, we thus have:

$$RMSLE = \sqrt{\frac{1}{n} \sum_{i=1}^{N} (\log(p_i + 1) - \log(a_i + 1))^2}$$

RMSLE is chosen because it only accounts to the percentual differences between predicted and actual values, not the absolute differences, as is with RMSE. So, in a way, RMSLE averages the square of the percentual differences between predicted and actual values.

## Benchmark model

For each calculated final model, sale prices will be predicted from the test set and the results will be submited to Kaggle, which will compare against the (unknown to me) real sale prices and calculate its RMSLE score.

Kaggle provides a sample submission file for the competition. I submited it with no modifications, and it scored *0.40890*. My first aim is to do better than that.

The competition's leaderboard provides the best scores from other competitors, so it can be used as a second benchmark to see how well my models fare against the other competitors.

## Project design

- Features (input variables) will be thoroughly examined, by themselves and against the target variable (sale price), in order to do some feature engineering. Features may be removed, modified or even added, with a close eye to the treatment of missing values. Special attention will be held for categorical variables, which may be codified in some way to be turned into numerical variables.
    - Features with small correlation to the target variable, or which contribute very little to the regression score, may be removed. One expected benefit is to avoid overfitting.
    - One possible way to deal with categorical variables it to use *one-hot encoding*. That is: if variable $A$ has $A_1, \ldots, A_n$ as possible values, then we create $n-1$ binary variables named $A.A_1, \ldots, A_{n-1}$ where at most one of them will be valued as $1$, according to $A$'s value. However, if $A = A_n$, then all of them will be zero.
    - The target variable may also be engineered, as suggested in the “Dataset and inputs” section.
- I consider using 4 regression models available in the scikit-learn Python library: Linear Regression, Random Forrest, Gradient Boosting Trees and Support Vector Machine.
- Model parameters will be tuned with help of a grid search technique, with k-fold cross validation using subsets of the training set as validation sets.
- For each different regression model, after I consider the model is fit to its best, I will predict over the test set and submit to Kaggle, and record and report the attained score. My score will be readily compared to other competitor's scores as benchmark.
- In the end, I will combine my best models and average them, with help of cross validation to assess whether it is a good idea or not, and submit to Kaggle to hear the truth, so to speak.