Udacity Machine Learning Engineer Nanodegree

Capstone Project

Amrish Purohit July 07, 2018

Table of Contents

[1. Definition 2](#_Toc518861524)

[1.1 Project Overview 2](#_Toc518861525)

[1.2. Problem Statement 2](#_Toc518861526)

[1.3 Datasets and Inputs 3](#_Toc518861527)

[1.4. Evaluation Metrics 3](#_Toc518861528)

[2. Analysis 4](#_Toc518861529)

[2.1. Data Exploration 4](#_Toc518861530)

[2.1.1 Dataset files 4](#_Toc518861531)

[2.1.2 Samples and shapes of the dataset 4](#_Toc518861532)

[2.1.3 Statistics of dataset 5](#_Toc518861533)

[2.1.4 Features with higher missing values 5](#_Toc518861534)

[2.1.5 Correlated features 6](#_Toc518861535)

[2.2. Exploratory Visualizations 6](#_Toc518861536)

[2.3. Algorithms and Techniques 8](#_Toc518861537)

[2.4. Benchmark 10](#_Toc518861538)

[3. Methodology 10](#_Toc518861539)

[3.1. Data Preprocessing 10](#_Toc518861540)

[3.2 Feature Engineering. 12](#_Toc518861541)

[3.2. Implementation 14](#_Toc518861542)

[3.2.1. Cross-Validation 15](#_Toc518861543)

[3.3. Refinement 15](#_Toc518861544)

[4. Results 16](#_Toc518861545)

[4.1 Model Evaluation and Validation 16](#_Toc518861546)

[4.2 Justification 17](#_Toc518861547)

[4.3 2017 Data 17](#_Toc518861548)

[5. Conclusion 18](#_Toc518861549)

[5.1 Free-Form Visualization 18](#_Toc518861550)

[5.2 Reflection 19](#_Toc518861551)

[5.3 Improvements 19](#_Toc518861552)

[5.4 References 20](#_Toc518861553)

# **1. Definition**

## **1.1 Project Overview**

Estimating price of a house is quite challenging and important process as housing prices are difficult to predict, and can be influenced by a very large quantity of factors. Everyone want better home in lease amount. This is lifetime decision and people spends hundreds of thousands of dollars to buy a house (at least in Los Angeles and major cities of California and US). To make such important decision, buyer want to make sure that fair price is placed on the property, particularly price of the house is not inflated.

Now days there are many real estate companies provide data and own define algorithm to determine best price of the house. One of such company is Zillow. Zillow has millions of data on homes across United States. Zillow has machine learning algorithm called “Zestimate”. The Zestimate home value is Zillow's estimated market value for an individual home and is calculated for about 100 million homes nationwide. The Zestimate is calculated from public and user-submitted data, considering special features, location, and market conditions. More information about Zestimate can be obtained from here [Zestimate](https://www.zillow.com/zestimate).

## **1.2. Problem Statement**

Based on data set provided on Kaggle, <https://www.kaggle.com/c/zillow-prize-1>, goal of the project is to predict price of new property going to sold in Los Angeles, Orange and Ventura county of California. Here price of home is evaluated based on features of home, it is a regression problem.

As this is supervised regression problem, challenge here is to predict target variable logerror. Logerror is defined as logerror = log(Zestimate)−log(SalePrice). To approach this problem I will be using various tree based regression algorithms as they are easy to work with, relatively fast trainers, somewhat robust to the outliers and random noise present in the data due to their ensemble nature.

As a benchmark model I will use simple ‘Linear regression’ and for final implementation I will use various regression algorithms like BayesianRidge, Ridge, Lasso, DecisionTreeRegressor, RandomForestRegressor, KNeighborsRegressor, XGBRegressor, GradientBoostingRegressor, CatBoostRegressor. All these algorithms are executed with default parameters and choose one best model based on score yield by these models. 'neg\_mean\_absolute\_error’ is configured to define score of the model with Kfold value of 10. Finally, XGBRegressor is chosen as final model and tuned with GridSearchCV object.

## **1.3 Datasets and Inputs**

For this project, I will use data set provided by Zillow on Kaggle competition, <https://www.kaggle.com/c/zillow-prize-1>. The train data has all the transactions before October 15, 2016, plus some of the transactions after October 15, 2016 and test data in the public leaderboard has the rest of the transactions between October 15 and December 31, 2016. About 58 features provided in properties file like size, neighborhood, tax and location.

The data set will be used to train model to estimate price of home and prediction will be compared with Kaggle board.

## **1.4. Evaluation Metrics**

Submissions are evaluated on Mean Absolute Error between the predicted log error and the actual log error. The log error is defined as

*logerror=log(Zestimate)−log(SalePrice)*

**Mean Absolute Error (MAE)** is a measure of difference between two continuous variables. Assume *X* and *Y* are variables of paired observations that express the same phenomenon. Examples of *Y* versus *X* include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. Consider a scatter plot of *n* points, where point *i* has coordinates ( *xi*, *yi* ).

Mean Absolute Error (MAE) is the average vertical distance between each point and the Y=X line, which is also known as the One-to-One line. MAE is also the average horizontal distance between each point and the Y=X line. The solution for the problem will be written in python using scikit and other packages. As, the problem is a regression problem and because the evaluation of the solution is based on MEA, ‘neg\_mean\_absolute\_error’ is chosen for model evaluation. This ‘regression’ scoring metric provided by scikit is used to measure the MAE value which is used to measure the Zillow problem. MAE output is non-negative floating point.

The metric measures the distance between the model and the data and in case of ‘neg\_mean\_absolute\_error’, the lower return values are better than the higher return values. The best value is 0.0.

# **2. Analysis**

## **2.1. Data Exploration**

The provided dataset is a tabular, comma separated values sheet with nearly 3 million samples that have 58 features to describe them. It is actual real estate information from 3 counties (Los Angeles, Orange and Ventura) in California. Zillow’s data is obtained from publicly available sources, and since different cities and counties track statistics in different ways

### **2.1.1 Dataset files**

We are provided with two sets of training data and their corresponding “properties” file: (i) “train\_2016\_v2.csv” and "properties\_2016", (ii) “train\_2017.csv” and "properties\_2017". The main difference between "properties\_2016" and "properties\_2017" are in tax features.

### **2.1.2 Samples and shapes of the dataset**

Here is sample of **train\_2016\_v2**

parcelid logerror transactiondate

0 11016594 0.0276 2016-01-01

1 14366692 -0.1684 2016-01-01

2 12098116 -0.0040 2016-01-01

3 12643413 0.0218 2016-01-02

4 14432541 -0.0050 2016-01-02

Shape of the set is (90275, 3)

Shape of properties file (2985217, 58). As this data is very big, I cannot present sample here. Please refer **“Read and Prepare Dataset”** section of corresponding **project.ipynb** file.

Shape Of Merged Training Data is (90275, 60)

### **2.1.3 Statistics of dataset**

|  |
| --- |
| parcelid logerror airconditioningtypeid architecturalstyletypeid basementsqft bathroomcnt bedroomcnt \ |
| count 9.028e+04 90275.000 28781.000 261.000 43.000 90275.000 90275.000 |
| mean 1.298e+07 0.011 1.816 7.230 713.581 2.279 3.032 |
| std 2.505e+06 0.161 2.974 2.716 437.434 1.004 1.156 |
| min 1.071e+07 -4.605 1.000 2.000 100.000 0.000 0.000 |
| 25% 1.156e+07 -0.025 1.000 7.000 407.500 2.000 2.000 |
| 50% 1.255e+07 0.006 1.000 7.000 616.000 2.000 3.000 |
| 75% 1.423e+07 0.039 1.000 7.000 872.000 3.000 4.000 |
| max 1.630e+08 4.737 13.000 21.000 1555.000 20.000 16.000 |
|  |
| buildingclasstypeid buildingqualitytypeid calculatedbathnbr ... yardbuildingsqft26 \ |
| count 16.0 57364.000 89093.000 ... 95.000 |
| mean 4.0 5.565 2.309 ... 311.695 |
| std 0.0 1.901 0.976 ... 346.355 |
| min 4.0 1.000 1.000 ... 18.000 |
| 25% 4.0 4.000 2.000 ... 100.000 |
| 50% 4.0 7.000 2.000 ... 159.000 |
| 75% 4.0 7.000 3.000 ... 361.000 |
| max 4.0 12.000 20.000 ... 1366.000 |
|  |
| yearbuilt numberofstories structuretaxvaluedollarcnt taxvaluedollarcnt assessmentyear \ |
| count 89519.000 20570.000 8.990e+04 9.027e+04 90275.0 |
| mean 1968.533 1.441 1.801e+05 4.577e+05 2015.0 |
| std 23.763 0.544 2.091e+05 5.549e+05 0.0 |
| min 1885.000 1.000 1.000e+02 2.200e+01 2015.0 |
| 25% 1953.000 1.000 8.124e+04 1.990e+05 2015.0 |
| 50% 1970.000 1.000 1.320e+05 3.429e+05 2015.0 |
| 75% 1987.000 2.000 2.105e+05 5.406e+05 2015.0 |
| max 2015.000 4.000 9.948e+06 2.775e+07 2015.0 |
|  |
| landtaxvaluedollarcnt taxamount taxdelinquencyyear censustractandblock |
| count 9.027e+04 90269.000 1783.000 8.967e+04 |
| mean 2.783e+05 5983.976 13.403 6.049e+13 |
| std 4.005e+05 6838.877 2.716 2.047e+11 |
| min 2.200e+01 49.080 6.000 6.037e+13 |
| 25% 8.223e+04 2872.830 13.000 6.037e+13 |
| 50% 1.930e+05 4542.750 14.000 6.038e+13 |
| 75% 3.454e+05 6901.090 15.000 6.059e+13 |

Target variable logerror ranges from -4.605 to 4.737

### **2.1.4 Features with higher missing values**

When performing missing data analysis, we can observe that most of the variables have a high ratio of missing data. There are 25 variables which have a missing ratio of > 70%.

* architecturalstyletypeid
* basementsqft
* buildingclasstypeid
* decktypeid
* finishedfloor1squarefeet
* finishedsquarefeet13
* finishedsquarefeet15
* finishedsquarefeet50
* finishedsquarefeet6
* fireplacecnt
* hashottuborspa
* poolcnt
* poolsizesum
* pooltypeid10
* pooltypeid2
* pooltypeid7
* storytypeid
* threequarterbathnbr
* typeconstructiontypeid
* yardbuildingsqft17
* yardbuildingsqft26
* numberofstories
* fireplaceflag
* taxdelinquencyflag
* taxdelinquencyyear

### **2.1.5 Correlated features**

From correlation analysis, we can observe that following pairs are highly correlated.

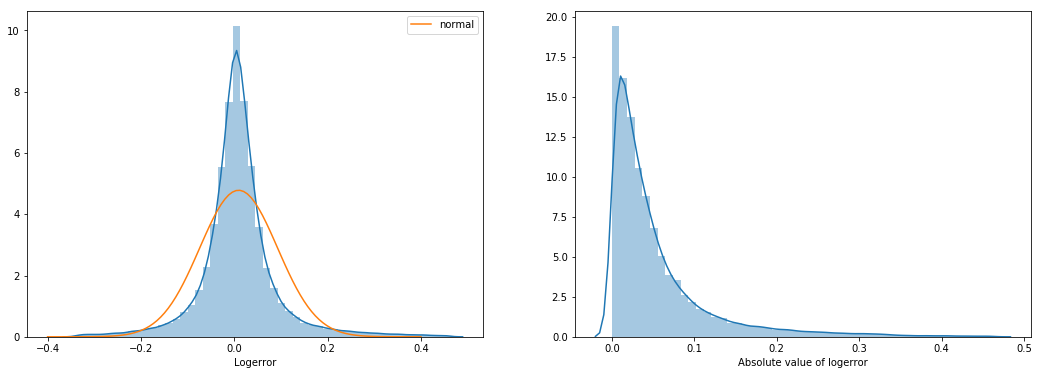
* (bathroomcnt, calculatedbathnbr and fullbathcnt)
* (calculatedfinishedsquarefeet and finishedsquarefeet12)
* (fips, rawcensustractandblock, censustrackandblock) and
* (taxvaluedollarcnt, taxamount, landvaluedollarcnt).

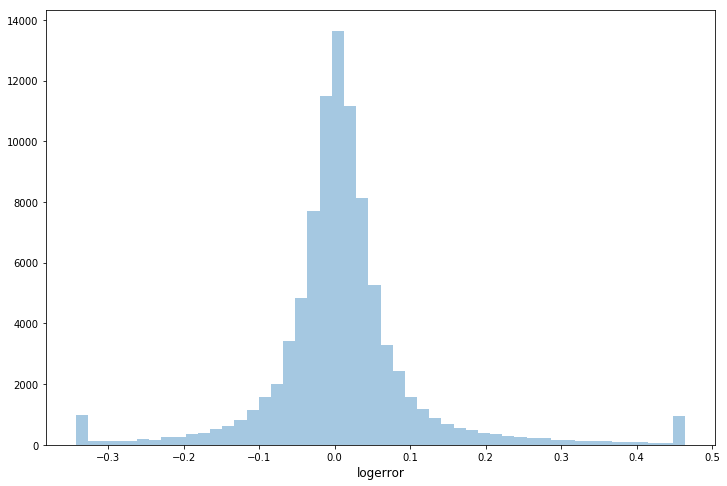
The data has been divided into continuous, discrete and categorical data. By plotting histograms on the continuous variables, we can observe that most of the data is normally distributed.

Preprocessing for this dataset will have to include alterations to the features like replacing missing values (where applicable) and changing the way some of the features are represented via encoding strategies.

## **2.2. Exploratory Visualizations**

The target variable ‘logerror’ has few outliers but is mostly normally distributed.

****



The data set "properties\_2016" consists of 58 features and 2,985,217 observations. The features include information about the size, neighborhood, tax, and location of the properties.

Data has been largely divided into three categories.

1. continuous

'taxamount','garagetotalsqft', 'calculatedfinishedsquarefeet', 'lotsizesquarefeet','structuretaxvaluedollarcnt'

continuous variables are depicted through display plot

2. discrete

'bathroomcnt', 'bedroomcnt','garagecarcnt', 'roomcnt', 'unitcnt','assessmentyear', 'yearbuilt'

discrete variables are depicted through count plot

3. categorical

'airconditioningtypeid', 'buildingqualitytypeid', 'heatingorsystemtypeid', 'propertylandusetypeid', 'fips', 'regionidcounty'

categorical variables are depicted through bar plot.

## **2.3. Algorithms and Techniques**

Initially I was planning to use deep learning technique **CNN** to implement the algorithm as I am very impressed with deep learning during course of the program. As feature set is relative small in context of deep learning and comment I got from my connect program teacher, It may tend to over fitting on training set.

Intention here is to use algorithms that I have learnt and try to pick one which provides higher score. Here is list of algorithms used and brief detail.

* **BayesianRidge** - Bayesian linear regression is an approach to linear regression in which the statistical analysis is undertaken within the context of Bayesian inference. Like current dataset, it is used when the regression model has errors that have a normal distribution
* **Ridge** - Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity (correlations between predictor variables). It is a remedial measure taken to alleviate multicollinearity amongst regression.
* **Lasso** - LASSO (Least Absolute Shrinkage Selector Operator) regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models This type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.
* BayesianRidge, Ridge and Lasso are variant of chosen benchmark model Linear regression.
* **DecisionTreeRegressor** - Decision tree builds regression or classification models in the form of a tree structure. It brakes down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.
* **RandomForestRegressor** – Random forest regressor is variant of above mentioned decision tree regressor. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.
* **KNeighborsRegressor** - k-nearest neighbors algorithm (k-NN) is a no In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. n-parametric method used for classification and regression.
* **XGBRegressor** - XGBoost is short for “Extreme Gradient Boosting” is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost is the leading model for working with standard tabular data. XGBoost belongs to a family of boosting algorithms that convert weak learners into strong learners.
* **GradientBoostingRegressor** - Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.
* **CatBoostRegressor** – CatBoostRegressor is also gradient boosting algorithm. CatBoost is able to incorporate categorical features in data like music genre, device id, URL, etc in predictive models with no additional preprocessing. It is mostly used for categorical data which is not easy build up in decision tree.
* I chose set of regression model which mix of linear and decision tree based boosting algorithms.

I run all the model with default parameter and XGBoost came ups with highest score. I chose XGBoost as final model to tune it. XGBoost is an ensemble method that makes predictions by combining the output of several trees. XGBoost is fast compared to other implementations of gradient boosting. XGBoost is very good with structured or tabular datasets on classification and regression predictive modeling problems.

XGBoost optimizes an objective function. The objective function is a summation of a loss function and a regularization function which depends on the leaf weights and the number of leaves in a tree.

Below are the main parameters that was used to optimize the model in this project:

Nthread = To utilize all cores of the processor

max\_depth (9) = the maximum depth of each tree.

min\_child\_weight (100) = the minimum number of samples that must be present in each node

n\_estimators (100) = Number of boosted trees to fit.

learning\_rate (0.1) = this parameter scales down the steps along the gradient. It can be used to make the boosting process more conservative.

colsample\_bytree (1) = the ratio of the features that is used to construct a tree.

Subsample (0.8) = ratio of the training data points that are randomly selected to construct each tree.

Once the base score is achieved by the model, I have tuned the same by using Grid Search and Cross Validation methods.

## **2.4. Benchmark**

As I have mentioned in my proposal, I have used Linear regression as benchmark model. Linear regressor got score of -0.06850. I have used other tree base ensemble methods. I have observed that XGBoost provides better result.

# **3. Methodology**

## **3.1. Data Preprocessing**

preliminary assessment of the dataset for missing values is required before fitting data into the model as dataset contains features with lots of missing values more than 70%.

Merge training data file (train\_2016\_v2.csv) and properties file(properties\_2016.csv).

Cleaning data by dropping columns which have a missing ratio of more than 70%. Here is list variable dropped due to missing values

* architecturalstyletypeid,
* basementsqft
* buildingclasstypeid
* decktypeid
* finishedfloor1squarefeet
* finishedsquarefeet13
* finishedsquarefeet15
* finishedsquarefeet50
* finishedsquarefeet6
* fireplacecnt
* hashottuborspa
* poolcnt
* poolsizesum
* pooltypeid10
* pooltypeid2
* pooltypeid7
* storytypeid
* threequarterbathnbr
* typeconstructiontypeid
* yardbuildingsqft17
* yardbuildingsqft26
* numberofstories
* fireplaceflag
* taxdelinquencyflag
* taxdelinquencyyear

On correlation analysis, it was observed that the following pairs are highly correlated.

* fips, rawcensustractandblock, censustrackandblock
* taxvaluedollarcnt, taxamount, landvaluedollarcnt
* bathroomcnt, calculatedbathnbr and fullbathcnt
* calculatedfinishedsquarefeet and finishedsquarefeet12

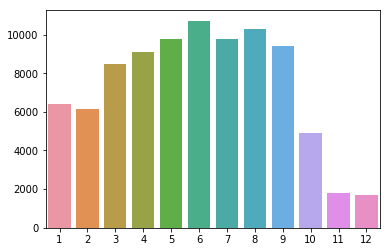
Among above variables, variable with high missing value is dropped.

Numeric variables with null columns, data was imputed with median values.

Variables with null columns, data was imputed with -1.

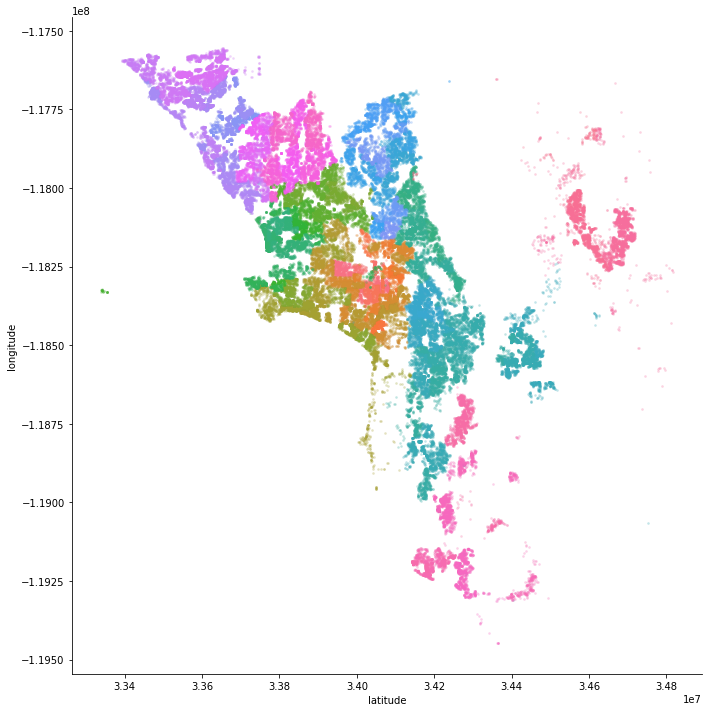
Outliers were eliminated from target variable ‘log\_error’.

## **3.2 Feature Engineering.**

A bar plot is generated to understand transactions per month.  


New\_Transaction\_Month column is added to the training model.

New\_zip\_count : Number of properties in the zip



New\_LivingAreaProp : Proportion of living area is added.

New\_city\_count : Number of properties in the city is added.

## **3.2. Implementation**

* **Packages and Dataset :** I have used Jupyter notebooks with Python3 for implementation. Libraries and packages (numpy, pandas, matplotlib, seaborn, sklearn, catboost, xgboost) are imported prior to implementation and number of CPU cores are calculated with multiprocessing package. Other packages are imported as needed.
* **Benchmark Model:** Initial benchmark model is created with Linear regression as already mentioned in proposal. Linear regression yield score of 0.06850.
* **Read and prepare dataset:** Read and merge training data and properties data on unique field ‘parcelid’. Columns are dropped where Nan values are more than 70%. Null values are imputed and median values are replaced in numeric column
* **Correlation Analysis:** Correlation between features are plotted with heat map and columns with high missing values are dropped from the training set.
* **Removing Outliers:** Outliers are plotted with target variable and removed from training data set.
* **Feature Engineering :** New feature as described in previous section is added with feature engineering.
* **Model Selection:** I have tried following models with default parameters.

|  |  |
| --- | --- |
| Model Name | Score |
| Bayesian Ridge Regression | -0.05314 |
| Linear Regression | -0.05315 |
| Ridge Regression | -0.05315 |
| Lasso Regression | -0.05314 |
| Decision Trees | - 0.08308 |
| Random Forest | -0.05925 |
| KNN | -0.06117 |
| XGB Regression | -0.05280 |
| Gradient Boosted Regressor | -0.05285 |
| CatBoostRegressor Regressor | -0.05280 |

Among above models, XGB Regression is selected with score -0.052796701349816476.

* **Model tuning :** Selected model is tuned with GridSearchCV. Detail is provided in next refinement section.

### **3.2.1. Cross-Validation**

I have used used the k-fold cross-validation scheme with k=10. In k-fold cross-validation method, we train a model k times on (1-1/k)\*100 percent of the training data.

The remaining 1/k\*100 percent of the data is used for validation. The validation dataset is randomly selected from the training dataset. Those k models are then used to make a prediction for the test dataset.

After cross validation final score obtained from tuned model is Final score : -0.05277.

## **3.3. Refinement**

Benchmark Model Linear regression provide accuracy of -0.06850. This score was based before preprocessing of the data. After preprocessing, cleaning and feature engineering on data. I ran Linear regression again and yield score of -0.05315. I ran all the model described above and chose XGB regressor as a final model.

I created the GridSearchCV object to tune the parameters for the XGB regressor. Parameters that I have considered for tuning are as follow.

*params = {*

*"nthread" : [ncpu],*

*"max\_depth": [3,4,5,7,9],*

*"min\_child\_weight": [1,10,50,100],*

*"n\_estimators": [10,100,500,1000],*

*"subsample" : [0.5, 0.8, 1],*

*"colsample\_bytree" : [0.5, 1],*

*"learning\_rate":[0.1, 0.01]*

*}*

I ran grid search object for hours with permutations and combinations to find out right parameters. Final parameters chosen are as follow, output from Grid Search.

**Best Score : -0.052802641831849545**

**Best Parameters : {'colsample\_bytree': 1, 'learning\_rate': 0.1, 'max\_depth': 3, 'min\_child\_weight': 100, 'n\_estimators': 100, 'nthread': 8, 'subsample': 0.8}**

**Best Estimator : XGBRegressor(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,**

**colsample\_bytree=1, gamma=0, learning\_rate=0.1, max\_delta\_step=0,**

**max\_depth=3, min\_child\_weight=100, missing=None, n\_estimators=100,**

**n\_jobs=1, nthread=8, objective='reg:linear', random\_state=0,**

**reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=1, seed=None,**

**silent=True, subsample=0.8)**

# **4. Results**

## **4.1 Model Evaluation and Validation**

After testing various liner regression and decision tree based ensemble models, I chose XGBoost regressor.

XGBoost with default parameters yield score of -0.05280. XGBoost has a few parameters that can affect model's accuracy and training speed. XGBoost parameters are listed on <https://xgboost.readthedocs.io/en/latest/python/python_api.html>. I choose following parameters for tuning.

params = {

"nthread" : [ncpu],

"max\_depth": [3,4,5,7,9],

"min\_child\_weight": [1,10,50,100],

"n\_estimators": [10,100,500,1000],

"subsample" : [0.5, 0.8, 1],

"colsample\_bytree" : [0.5, 1],

"learning\_rate":[0.1, 0.01]

}

I created the GridSearchCV object to tune the parameters for the XGB regressor. I sept couple hours to choose parameters which yield better scores. The first parameter that affect the performance is *n\_estimators*. n\_estimators specifies how many times to go through the modeling cycle described above. Too low a value causes underfitting, which is inaccurate predictions on both training data and new data. Too large a value causes overfitting, which is accurate predictions on training data, but inaccurate predictions on new data. After many iterations n\_estimators, value chosen is 100.

learning\_rate is another important feature, I have tried learning rate from 0.1 and 0.01. After several iterations, learning 0.1 yield better results with other parameters.

Like n\_estimators and learning\_rate, I have tried other parameters mentioned above and here is list of parameters with final values which yield better score for this model.

*max\_depth=3*

*min\_child\_weight=100*

*n\_estimators=100*

*learning\_rate=0.1*

*colsample\_bytree=1*

*subsample=0.8*

The robustness of the model and its solution were tested by using Cross Validation with number of folds as 10. When the cv argument is an integer, cross\_val\_score uses the KFold or StratifiedKFold strategies by default.

Cross validation technique is generally used to assess the predictive ability of any regression model. The concept of cross-validation relies on the principle that a large enough dataset can split into two or more sub-groups, the first being used to derive the model and the additional data set(s) reserved for model testing and validation. This would let us know when our model might be over or under fitting on the dataset that we have employed.

## **4.2 Justification**

The final model scored the following on the CV and testing sets:

Benchmark Score = -0.06850

Final score after Model tuining = -0.05277

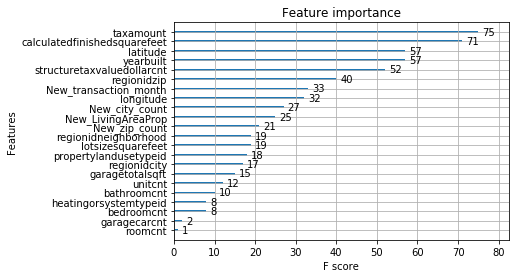
This is an improvement over the benchmark model. Steps followed in getting the result add significance and validity to the final solution obtained. Difference between Linear regression model and tuned XGB regressor obtained through data processing, removing outliers and feature engineering.

## **4.3 2017 Data**

The 2017 data were released on October 2, 2017. This data included transactions from January through mid-September, 2016. The teams were also provided with properties\_2017 dataset. The 2017 dataset has the same parcel\_id as in the properties\_2016 file but the tax assessment values are updated based on the data that were collected in 2017. The features that we engineered for the 2016 data can seamlessly be applied to the 2017 data without any needs for further adjustments.

# **5. Conclusion**

## **5.1 Free-Form Visualization**



The XGBoost library provides a built-in function to plot features ordered by their importance.

The function is called plot\_importance() which maps features to corresponding F score. The above mentioned diagram is obtained from using the plot\_importance() feature of XGBoost. Based on domain knowledge of the Housing Industry I can state that the features generated by the model are in sync with the real world housing market. For example, Most of the home buyers are interested in paying low tax amounts, look for homes with high calculatedfinishedsquarefeet, are interested in location of the house (latitute, longitude), age of the house - depicted by year built etc. All these features have been correctly marked as important by the model.

## **5.2 Reflection**

The following process was used followed to create this project.

* **Import Packages**
* **Benchmark Model - Linear Regression**
* **Read and Prepare Dataset**
* **Correlation Analysis**
* **Remove Outliers**
* **Univariate analysis**
* **Bivariate analysis**
* **Feature Engineering**
* **Model Selection**
* **Model Tuning**
* **Cross Validation**

Many of the above steps were iterative, as I made small adjustments a number of times to see if better scores could be achieved. I enjoyed and learnt a lot by exploratory data analysis of data, and application of different regression models and techniques such as gridsearch and cross-validation.

This capstone project helped me a lot to sharpen my knowledge and skills that I have learnt during the Udacity course.

## **5.3 Improvements**

* During model selection process, 1 out of 4 times, CatBoostRegressor was chosen as best model. I think CatBoostRegressor is very close match to XGB regressor. CatBoostRegressor could yield better result than XGB if we tune it.
* I could use KNN to impute missing values to get better result on final score.
* As I mentioned in proposal, I was wanted to try CNN to implement this project. CNN can be good candidate for model selection.
* One can use PCA and ICA to combine and transform some of the numerical variables into a low-dimensional space.

## **5.4 References**

* <https://www.kaggle.com/c/zillow-prize-1>
* <https://github.com/dmlc/xgboost>
* <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor>
* <https://en.wikipedia.org/wiki/Feature_engineering>