 # Machine Learning Engineer Nanodegree

Capstone Project

Adapa Janardhana Swamy

September 16th, 2017

Ref: https://www.kaggle.com/c/zillow-prize-1

Zillow's Home Value Prediction (Zestimate) – Kaggle Competition

I. Definition

Project Overview

The Zestimate is a predicted valuation of home by Zillow, which was created to give consumers as much information as possible about homes and the housing market, marking the first-time consumers had access to this type of home value information at no cost. Zillow has since become established as one of the largest, most trusted marketplaces for real estate information in the U.S. and a leading example of impactful machine learning. Now, I am trying to improve the accuracy as part of kaggle Competition. Kaggle provides the dataset for the transactions of selling the house happened in 2016 year with which I have worked on.

Problem Statement

I have predicted the logerror of the Zestimate which has been estimated based on features like- area, no.of bedrooms, region, and other facilities available. The housing price can be a linear combination of different features like – no.of bedrooms, no.of bathrooms, area in square feet, etc. We can measure our model's performance by comparing the predicted logerror with actual logerror.

Here, log error is between previous zestimate and actual sale price.

$$\text{logerror} = \log(\text{Zestimate}) - \log(\text{SalePrice})$$

This model can be used for future house sales also, in this competition we will predict for Oct-Dec2016 and for Oct-Dec2017.

Initially I have analyzed the data by visualizing between different features, and finding features which have correlation with target variable which is log error in our problem. I have tried multiple ML algorithms like linear regression, decision tree regressor, xgboost and I chose on which gave the best score. With the final model, if we pass the future data of the features, then we can predict the future prices also, which makes the model reproducible.

Metrics

I am using Mean Absolute Error between the predicted log error and the actual log error for the evaluation.

The mean absolute error is given by

$$MAE=1n\sum i=1n|yi-yi\hat{}|$$

Where yi is the actual value and the yi^ is the predicted value.

I have chosen MAE over RMSE because I want the metric to be insensitive to the outliers. RMSE will give more weight to large errors. I want the metric to be uniform for all ranges of errors, since we are predicting the logerror of the actual sale price.

This is also the evaluation metric used in the kaggle competition for the leaderboard ranking.

II. Analysis

Data Exploration

I have obtained the dataset from the Kaggle competition site(https://www.kaggle.com/c/zillow-prize-1/data) . The dataset has around 60 features like –

- type of cooling system present in home
- no.of bedrooms
- no.of bathrooms
- total area
- no.of pools

These features are explained in the file – Zillow_data_dictionary in the above link and the values for these are present in properties_2016.csv file whereas the train_2016_v2.csv contains homeid, transaction date (selling data), and the log error.

Sample of 'train_2016_v2.csv',

parcelid	logerror	transactiondate
11016594	0.0276	01-01-16
14366692	-0.1684	01-01-16
12098116	-0.004	01-01-16

Sample of 'properties_2016.csv',

parcelid	airconditioningtypeid	bathroomcnt	bedroomcnt	buildingqualitytypeid	calculatedfinishedsquarefeet	finishedsquarefeet12	fips	fireplacecnt
10754147	0.0	0.0	0.0	7.0	1572.0	1539.0	6037.0	0.0
10759547	0.0	0.0	0.0	7.0	1572.0	1539.0	6037.0	0.0
10843547	0.0	0.0	0.0	7.0	73026.0	1539.0	6037.0	0.0
10859147	0.0	0.0	0.0	7.0	5068.0	1539.0	6037.0	0.0
10879947	0.0	0.0	0.0	7.0	1776.0	1539.0	6037.0	0.0

In our dataset, the target variable – 'logerror' is a continuous variable. The features contain categorical variables like – bathroom count, bedroom count, pool count, garage count, and the continuous variables like latitude, longitude, total area in square feet.

Features - ['architecturalstyletypeid', 'basementsqft', 'finishedsquarefeet13', 'storytypeid', 'typeconstructiontypeid', 'yardbuildingsqft26', 'fireplaceflag'] have non-null values less than 10000 whereas the total no.of samples is 90000.

Stats of a few features in our dataset -

	bathroomcnt	bedroomcnt	calculatedfinishedsquarefeet	yearbuilt
count	90275.000000	90275.000000	89614.000000	89519.000000
mean	2.279474	3.031869	1773.185987	1968.532870
std	1.004271	1.156436	928.162393	23.763475
min	0.000000	0.000000	2.000000	1885.000000
25%	2.000000	2.000000	1184.000000	1953.000000
50%	2.000000	3.000000	1540.000000	1970.000000
75%	3.000000	4.000000	2095.000000	1987.000000
max	20.000000	16.000000	22741.000000	2015.000000

Most of the samples have bathroom count as '2', bedroom count between 2-4, finished square feet between 1213-2136, and year in which the house is built is in-between 1950-1980.

Distribution of our target variable – 'logerror' –

```
plt.hist(data['logerror'],bins=10)
```

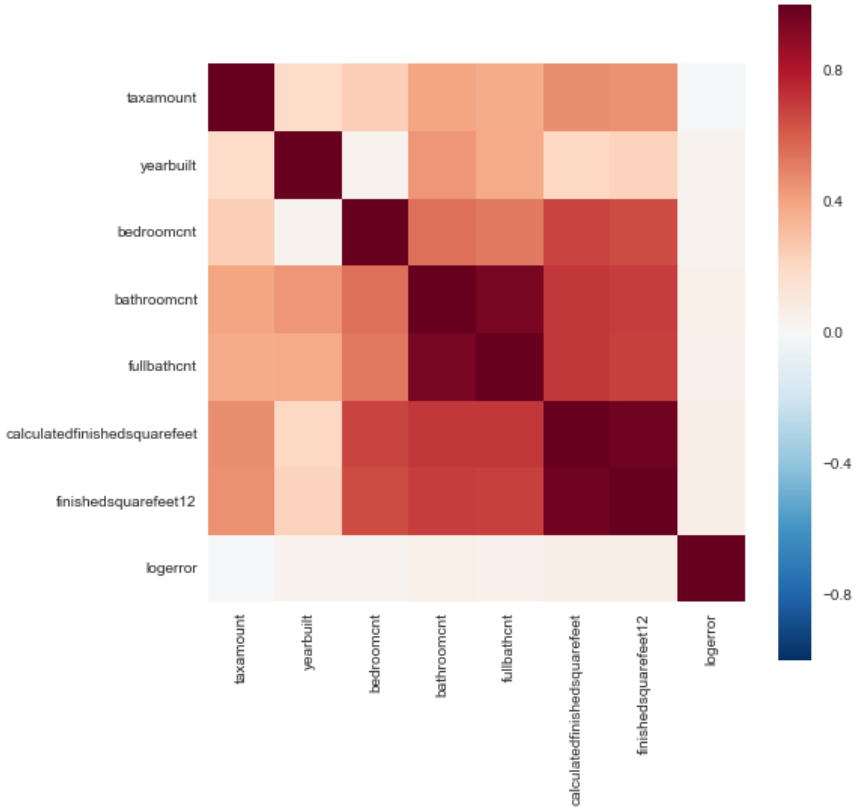
```
(array([  3.00000000e+00,   2.00000000e+00,   4.20000000e+01,
          6.50000000e+01,   7.47400000e+03,   7.73330000e+04,
          2.20000000e+02,   5.30000000e+01,   1.40000000e+01,
          2.00000000e+00]),
 array([-4.605, -3.7  , -2.795, -1.89 , -0.985, -0.08 ,  0.825,  1.73 ,
         2.635,  3.54 ,  4.445]),
 <a list of 10 Patch objects>)
```

We can consider logerror above 2.6 and below -2.7 as outliers as they have very low count.

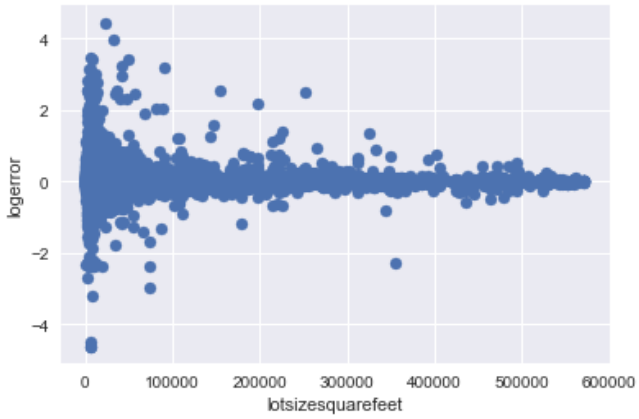
Exploratory Visualization

This is the correlation heat map between few features and logerror.

From the below plot, logerror is not directly correlated to any of the features, and fullbathcnt, bathroomcnt are highly correlated and calculatedfinishedsquarefeet, finishedsquarefeet12 are highly correlated. Here, finishedsquarefeet12 is finished living area and calculatedfinishedsquarefeet is calculated total finished living area. And we can also see that bedroomcnt, bathroomcnt, fullbathcnt are moderately correlated to finshedsquarefeet12, and calculatedfinishedsquarefeet.



Here is another example of no correlation between a feature and the target variable, this is the scatter plot between lotsizesquarefeet – area of lot in squarefeet and logerror -

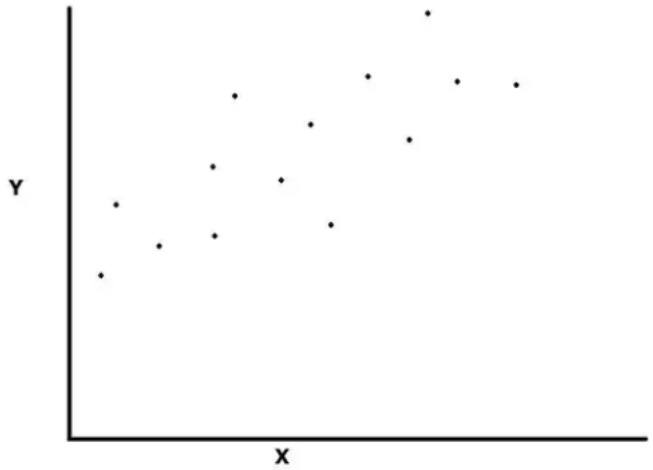


Algorithms and Techniques

I intend to use linear regression, and Xgboost as ML algorithms, and GridSearchCV to tune the parameters.

Linear regression model: In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Linear regression models are fitted using the least squares approach.

So, for example, in simple linear regression we have a bunch of points and we are trying to find a straight line with 'best fit' to accommodate them.
The equation of a straight line is, $Y = c + mX$
So the regression (line) equation can be written as, $Y = \beta 0 + \beta 1 X$
$\beta 0$ is the intercept and $\beta 1$ is the slope. Both are called model coefficients or model parameters.
$\beta 1$ represents the amount by which Y should change if we change X by one unit.
So basically building simple linear regression model is, solving this equation for values of model coefficients.

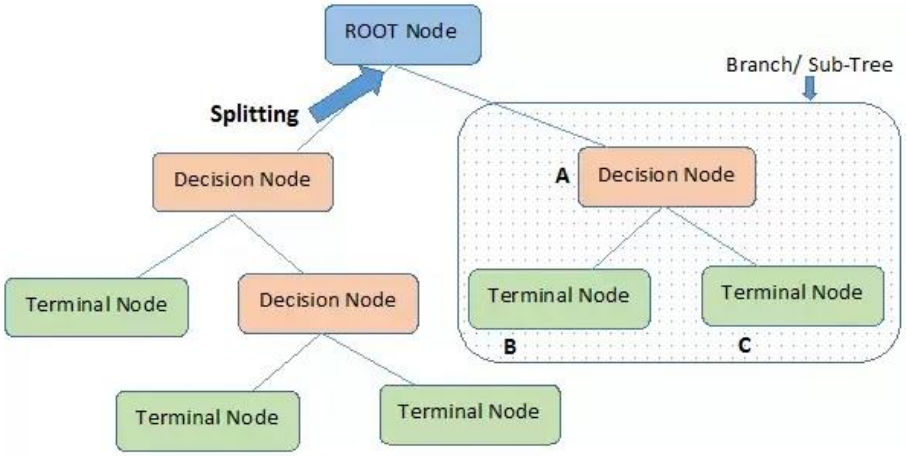


Now looking at the scatter plot above, it is impossible to fit a straight line through all data points. So, we modify the equation slightly.
$Y = \beta 0 + \beta 1 X + e$
Where "e" is the error while calculating Y.
Now let's get back to how do we decide the best fit. A regression line, which helps us to minimize the error component discussed in equation above, will be considered best fit. We will consider a regression line as a best fit, if it has the lowest sum of squares error.

Xgboost: XGBoost is short for "Extreme Gradient Boosting", Xgboost is an ensemble of trees. Usually, a single tree is not strong enough to be used in practice. What is used is the so-called tree ensemble model, which sums the prediction of multiple trees together.

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. This can be represented as graph. Now you can

parse through the possible nodes and edges as per the condition rules (if condition1 and condition2 and condition3 then outcome.)



Note:- A is parent node of B and C.

It is not easy to train all the trees at once. Instead, we use an additive strategy: fix what we have learned, and add one new tree at a time. We will add the one that optimizes our objective. Our objective is -

$$\text{obj} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i)$$

This uses MSE as loss function, ideally, we would enumerate all possible trees and pick the best one. In practice this is intractable, so we will try to optimize one level of the tree at a time. Specifically, we try to split a leaf into two leaves. For real valued data, we usually want to search for an optimal split. To efficiently do so, we place all the instances in sorted order. XGBoost is developed with both deep consideration in terms of *systems optimization* and *principles in machine learning*. The goal of this library is to push the extreme of the computation limits of machines to provide scalable, portable and accurate library.

GridSearchCV:

In the grid search technique, we can pass some parameters to this function and it will generate all possible combinations with those parameters and apply the learning algorithm to them. And it will return the optimal fit of the algorithm where we are getting the maximum score. We can use it to optimize an algorithm by passing the algorithm and some parameters to it, and it will give the optimal combination of parameters for which we will get the best score.

GridSearchCV will also do cross-validation. Cross Validation means it will divide the data into 'k' equal bins, and apply learning algorithm with one selected bin as test data, and remaining bins as training data, and then pick another bin for test data and so on. It will take the average of accuracies of K iterations and will return it. If we did not do CV and just used train-test split, the disadvantage is that the evaluation may highly depend on which data ends up in training set and which data ends up in testing

set. Whereas in doing CV, each data point gets to be in the testing set for exactly once. so, by doing Cross Validation we can ensure that the division of data doesn't affect the evaluation of the model.

Since the target variable – 'logerror' is a continuous variable, I cannot use classifying algorithms and I need to use regression ML algorithms for this problem. I intend to use linear regression as the training time is very low for it. I chose xgboost as it gives very good score as compared to linear regression and other ML algorithms. I chose gridsearch CV to tune the hyperparameters of the algorithm and to cross validate the dataset.

Benchmark

I have created a simple benchmark model for this problem –

I have used linear regression algorithm as my prediction model,

I have dropped the features - ['propertycountylandusecode','hashottuborspa', 'propertyzoningdesc', 'propertyzoningdesc', 'fireplaceflag', 'taxdelinquencyflag'] which have strings as values,

I have filled the missing data of a features with its median value.

I have iterated this linear regression model over different train_test splits with test size = 0.2,

And the mean MAE score obtained is '0.068901234',

this is the benchmark score that I will be referring to in this project.

III. Methodology

Data Preprocessing

I have dropped the features which have no.of non-null values as less than 10000.

I have dropped features - ['pooltypeid7', 'assessmentyear', 'propertycountylandusecode', 'propertyzoningdesc'], because of their unreasonable categorical values. For example, 'pooltypeid7' has only one value, and 'propertyzoningdesc' takes 5000 unique strings as values.

I have filled zeros in the place of missing values for the features - ['poolcnt', 'threequarterbathnbr', 'airconditioningtypeid'] because these are counts, and missing value can be replaced as '0' as they are filled with only non-zero values. And I have filled the missing values for other features with their median value.

I have removed the outliers manually for some categorical featuers - ['bathroomcnt'] <6.5, ['bedroomcnt'] <12, ['garagecarcnt'] <3, ['roomcnt'] <11, ['unitcnt'] <6, ['yearbuilt'] >1900

I have removed the top 0.5 percentile for the continuous variables - ['calculatedfinishedsquarefeet', 'finishedsquarefeet12', 'garagetotalsqft', 'lotsizesquarefeet', 'structuretaxvaluedollarcnt', 'taxvaluedollarcnt', 'landtaxvaluedollarcnt', 'taxamount']

I have removed logerror above 2.6 and below -2.7 as outliers, as they have very low count in the data exploration section

I have modified the feature ‘transactiondate’ as ‘month’ as we are predicting the logerror on monthly basis.

Implementation

Initially I have split the dataset into training and validation sets using ‘train_test_split ‘, with test_size =0.2, i.e., out of 90000 samples, 20% randomly picked samples will be validation set, and the remaining 80% samples will be training set.

I have used linear regression for our model and I have used gridsearchCV with no.of folds=20, to tune the linear regression model by trying two possible values of parameter 'normalize'.

Then I have tried xgboost algorithm for our model, and I have used these values for the parameters - params['eta'] = 0.02, params['objective'] = 'reg:linear', params['eval_metric'] = 'mae', params['max_depth'] = 4, params['silent'] = 1

I have loaded the training set and validation set into DMatrices, an passed these matrices to train the xgboost model.

d_train = xgb.DMatrix(X_train, label=y_train)

d_valid = xgb.DMatrix(X_test, label=y_test)

I have used evaluation metric of this model as ‘mae’ – mean absolute error as it is the desired evaluation metric.

watchlist = [(d_train, 'train'), (d_valid, 'valid')],

model = xgb.train(params, d_train, 10000, watchlist, early_stopping_rounds=100, verbose_eval=10)

I have created the watchlist and passed it to the xgboost, it will train on the training set, and test the model on the validation set, and it will improve the model by adding additional trees based on the validation set scores. The training will stop when there is it detects that there is no improvement in score on the validation set.

I have created two nested ’for’ loops to iterate over multiple values of the parameters – ‘eta’, and ‘max_depth’ of the xgboost model.

The challenging aspect I found is to perform sensitivity analysis to validate the model for a change in input data. I didn’t do this kind of analysis before, I solved this challenge by iterating the final model with change in the train_test splits (not fixing the random state), by not fixing the random state, with every iteration, the train_test_split will take new samples for testing set and training set.

Refinement

My initial linear regression model’s score on validation set is 0. 0661949154706, and my optimized Linear regression model with parameter ‘normalize’ as ‘True’ is 0.0661949149662.
These are scores that I got from during my tuning of parameter – ‘normalize’ by using ‘GridSearchCV’:

Parameter- normalize	True	False
Mean MAE on validation set	0.06702413	0.06702413

My initial xgboost's score on the validation set with the above parameters is 0.06595294971, and my optimized model with parameters- 'eta'= 0.03, and 'max_depth' =3 is 0.06582538.
I have tuned the parameters – 'eta', 'max_depth' by iterating them over two nested for loops.
These are scores I got during the iterations over multiple combinations of values to the parameters:

MAE on validation set	Parameter – 'max_depth'	3	4	5
Parameter – 'eta'				
0.01		0.06583435	0.06596830	0.0660529
0.02		0.06582647	0.06595294	0.0660431
0.03		0.06582538	0.06592521	0.0660464

IV. Results

Model Evaluation and Validation

I pick the final model as the optimized xgboost model with parameters 'eta'= 0.03, and 'max_depth' =3 as it has the best score. Initially I have tried the xgboost model by fixing some values to the parameters, and then I have tuned the parameters – 'eta', 'max_depth' by iterating multiple values over the 'for' loop. My final model – xgboost's parameter values are – eta = 0.03, objective = 'reg:linear', eval_metric = 'mae', max_depth = 3

I have performed sensitivity analysis by changing the training set and validation set- by iterating the trai n_test_split over a for loop by not fixing the random state. I have performed 10 iterations, and the resultant scores have mean '0.06655216' and have a standard deviation of '0.001095426' which is very l ow. So, we can deduct that with change in the input data, our model performs consistently.

Justification

My final model has the mean score - '0. 06655216' which is the MAE between the predicted log error and the given log error.

My benchmark model mean MAE score is '0.068901234'.

There is a 3.5% decrease in the mean MAE score from the benchmark model to my final model.

This is a significant improvement in score and we can say that this final model is a significant solution in solving our problem.

V. Conclusion

Free-Form Visualization

This is the feature importance graph of the features in our dataset which I got from the xgboost model.

```
xgb.plot_importance(model) #Feature Importance
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24072b00>
```

From the plot, we can see that structuretaxvaluedollarcnt, taxamount, calculatedfinishedsqaurefeet, latitude, regionidzip, longitude, bathroomcnt, month, are the most important features.

Reflection

1. Initially, I have done data exploration, and visualized features with target variable and in-between themselves.
2. I have removed some features based on the no.of non-null values, and other factors.
3. I have identified outliers and removed them.
4. I have filled the missing values with zeros and the corresponding median values as necessary
5. I have created a new variable 'month' which is the transaction month from the old feature – 'transactiondate'
6. I have split the dataset into training and validation sets
7. I have used linear regression model and tuned it using GridSearchCV
8. I have used xgboost and tuned some parameters by iterating over the possible values
9. Performed sensitivity analysis by randomly splitting the dataset into training and validation sets

I have found that there is no strong correlation between any feature and the target variable as challenging in this problem because we are not able to find a relationship between any feature and the target variable.

Another interesting aspect I found in this project is to iterate my final model over multiple train test splits to do sensitivity analysis.

Improvement

While tuning parameters for the xgboost model, I have used two nested for loops, instead we can use RandomSearchCV and tune over broader ranges.

We can improve our model by trying deep learning or implementing an "ensemble of ensembles" such as model stacking

References:

Kaggle compeitition overview: https://www.kaggle.com/c/zillow-prize-1

Dataset: https://www.kaggle.com/c/zillow-prize-1/data

https://www.quora.com/How-would-linear-regression-be-described-and-explained-in-laymans-terms#

http://xgboost.readthedocs.io/en/latest/model.html