

Original Article

Improve Zillow's Home Value Prediction Estimator (Zestimate)

Arun Raveendran Nair

Manager, Software Engineering Capital One LLC.

Received Date: 22 February 2025

Revised Date: 29 March 2025

Accepted Date: 21 May 2025

Project Overview : The capstone project is based on the Kaggle competition developed by Zillow Inc., the online real estate database company. To provide better values to its customers, Zillow provides an estimate of the home sale price, called Zestimate. Zestimate is very popular, because it provides first time consumers, information about the house and housing market at no cost.

“Zestimates” are estimated home values based on millions of statistical and machine learning models that analyze hundreds of data points on each property. By continually improving the median margin of error, from 14% at the onset to 5% today, Zillow has established itself as one of the largest and trusted online real estate database for the US market.

Similar house sale predictions have been solved using machine learning. Such an example is at: <https://web.stanford.edu/class/cs221/2017/restricted/p-final/ianjones/final.pdf>. In the project, selling prices of houses in King County, USA is predicted using a number of factors.

The current problem is statement is slightly different in that it does not attempt to predict the selling price, but the logerror for each record. Though the problem statements are different, usual methodologies employed in a machine learning model development, including the above, are applicable to this project.

Keywords: Zillow, Zestimate, Real Estate, Home Value Estimation, Machine Learning, Predictive Modeling, Logerror Prediction, Housing Market, Kaggle Competition, Statistical Models, Property Data Analysis, Capstone Project, Home Price Prediction, King County Housing Data, Model Evaluation.

Problem Statement

Due to the success of Zestimate, Zillow continually builds on its machine learning models to predict sale prices of houses. The project attempts to build a model to improve the Zestimate residual error. The residual error is a measure of the difference between the predicted value and the actual value, denoted as $e_i = y_i - x_i$

Zillow developed a model which can predict the sale price of a house, Zestimate, based on certain features it collects over the years. But Zestimate could be very different from the actual selling price of the house. Hence Zillow constantly tries to improve on the model for prediction. In this project, the residual error between Zestimate and actual Sale Price will be computed and the mean absolute error or MAE is calculated. In the ideal scenario, the Zestimate should be equal to Sale Price, which gives a MAE = 0.000. But in reality, this is not possible as Zestimate is different from Sale Price. Hence the task is to minimize the residual error as much as possible

For building a model, there are some preliminary steps – data preparation, choosing relevant models, training the data and fitting a model, validating the model built using metrics and finally testing the model. The key factor which helps to build a good model is understanding the data, and how the different features interact between themselves to affect the output or target.

I have studied the dataset and analyzed the following:

- Categorical and Numerical features – Categorical data cannot be used for building model. Hence these were converted to numerical codes for modeling.
- Missing Data – Rows with all data missing is deleted and columns are imputed with the mean values.
- Multicollinearity – Understand the correlation between variables, segregate and isolate highly correlated variables.

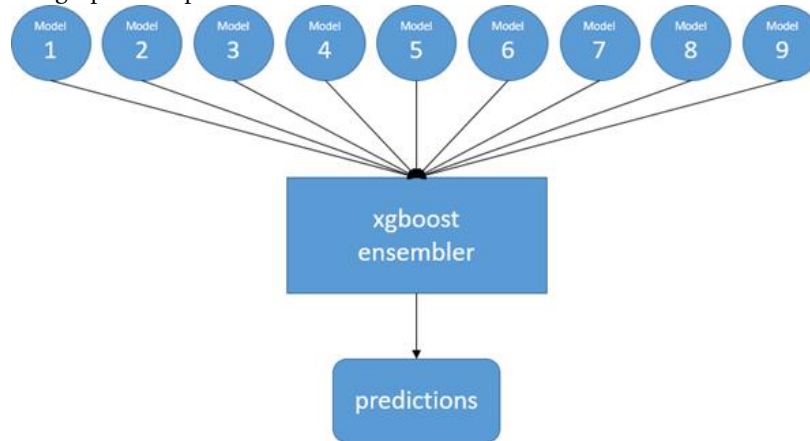
I build two models – Multiple Linear regression (MLR) and xgboost. My final model was a combined weighted model. The rationale behind choosing these two models is that MLR is a simple model of the form:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n + \epsilon$$



Where Y is the output and the X_1, \dots, X_n are the various predictors or features. ϵ is the residual error.

xgboost is a very complex model, developed by improving the computational capabilities of the more traditional Gradient Boosting Machine Model. The graphical depiction looks as follows:



Where each of Model 1, Model 2 etc. are different models and xgboost is an ensemble of these models. Hence the complexity is inherent.

Metrics: There are many different ways to measure the residual errors like MSE (Mean Squared Error), RMSE (Root Mean Squared Error), SSE (Sum of Squared Errors), MAE (Mean Absolute Error) etc. depending on the statistical analysis required.

In this project, the Mean Absolute Error (MAE) is used for evaluation. In statistical terminology, MAE is computed as follows:

$$MAE = \frac{\sum_{i=1}^n |y_i - x_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n}.$$

where y_i -> predicted value, x_i -> actual value, n -> number of observations.

Here, the mean absolute error between the predicted log error and the actual log error is considered. The log error is defined as:

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

and it is recorded in the transactions training data. If a transaction didn't happen for a property during that period of time, that row is ignored and not counted in the calculation of MAE.

A. Why MAE?

As mentioned earlier, though there are many methods to compute residual errors, the advantage of MAE for this project, is that it is more robust to outliers since it does not square the error. Other methods of residual computation employ squaring to prevent the positive and negative values from cancelling out each other and lowering the extend of differences. But in MAE, we take the absolute value and hence the effects of squaring do not occur. MAE is a more natural measure of average error.

II. ANALYSIS

A. Data Exploration:

The dataset given consists of the full list of real estate properties in three counties (Los Angeles, Orange and Ventura, California) data in 2016 and a few months' worth data from 2017. All data from 2016 is used to train the model and the model is tested on 2017 records.

The following files are provided:

- properties_2016.csv - all the properties with their home features for 2016.
- properties_2017.csv - all the properties with their home features for 2017.
- train_2016.csv - the training set with transactions from 1/1/2016 to 12/31/2016
- train_2017.csv - the training set with transactions from 1/1/2017 to 9/15/2017
- Data dictionary which describes the fields in more detail is available as a separate file - zillow_data_dictionary.xlsx.

A print of the data load step in Jupyter notebook, yields the following output:

```

Train Dataset has 90275 rows with 3 features each.
Train Dataset has 2985217 rows with 58 features each. Test Dataset has 30000 rows with
3 features each.
Test Dataset has 2985217 rows with 58 features each.
    
```

It can be seen that the properties files from the two years have the same number of records and features. Only the top 30000 records from 2017 are used in the test data.

A display of a few records from the train_2016 .csv file is as shown:

| parcelid | logerror | transactiondate | |
|----------|----------|-----------------|------------|
| 0 | 11016594 | 0.028 | 2016-01-01 |
| 1 | 14366692 | -0.168 | 2016-01-01 |
| 2 | 12098116 | -0.004 | 2016-01-01 |

A few features from the properties_2016 file looks as follows:

| parcelid | airconditioningtypeid | architecturalstyletypeid | basementsqft | bathroomcnt | bedroomcnt | buildingclasstypeid |
|----------|-----------------------|--------------------------|--------------|-------------|------------|---------------------|
| 0 | 1.10E+07 | NaN | NaN | NaN | 0 | 0 |
| 1 | 1.10E+07 | NaN | NaN | NaN | 0 | 0 |
| 2 | 1.10E+07 | NaN | NaN | NaN | 0 | 0 |

The descriptive statistics of these columns are as shown:

| | parcelid | logerr | aircondi | architec | baseme | bathroo | bedroo | buildi |
|-------|----------|--------|----------|----------|--------|---------|--------|---------------------|
| count | 90280 | ##### | 28781 | 261 | 43 | 90275 | 90275 | ngclasstypeid 16 |
| mean | 1E+07 | 0.01 | 1.816 | 7.23 | 713.58 | 2.279 | 3.032 | 4 |

| | | | | | | | | | |
|-----|------|-------|-------|-------|-------|--------|-------|-------|---|
| std | | 3E+06 | 0.16 | 2.974 | 2.716 | 437.43 | 1.004 | 1.156 | 0 |
| min | | 1E+07 | -4.61 | 1 | 2 | 100 | 0 | 0 | 4 |
| | 0.25 | 1E+07 | -0.03 | 1 | 7 | 407.5 | 2 | 2 | 4 |
| | 0.5 | 1E+07 | 0.01 | 1 | 7 | 616 | 2 | 3 | 4 |
| | 0.75 | 1E+07 | 0.04 | 1 | 7 | 872 | 3 | 4 | 4 |
| max | | 2E+08 | 4.74 | 13 | 21 | 1555 | 20 | 16 | 4 |

The features decryption from the data dictionary is given below. This helps to understand what each feature represents.

| Feature | Description |
|--------------------------------|---|
| 'airconditioningtypeid' | Type of cooling system present in the home (if any) |
| 'architecturalstyletypeid' | Architectural style of the home (i.e. ranch, colonial, split-level, etc...) |
| 'basementsqft' | Finished living area below or partially below ground level |
| 'bathroomcnt' | Number of bathrooms in home including fractional bathrooms |
| 'bedroomcnt' | Number of bedrooms in home |
| 'buildingqualitytypeid' | Overall assessment of condition of the building from best (lowest) to worst (highest) |
| 'buildingclasstypid' | The building framing type (steel frame, wood frame, concrete/brick) |
| 'calculatedbathnbr' | Number of bathrooms in home including fractional bathroom |
| 'decktypeid' | Type of deck (if any) present on parcel |
| 'threequarterbathnbr' | Number of 3/4 bathrooms in house (shower + sink + toilet) |
| 'finishedfloor1squarefeet' | Size of the finished living area on the first (entry) floor of the home |
| 'calculatedfinishedsquarefeet' | Calculated total finished living area of the home |
| 'finishedsquarefeet6' | Base unfinished and finished area |
| 'finishedsquarefeet12' | Finished living area |
| 'finishedsquarefeet13' | Perimeter living area |
| 'finishedsquarefeet15' | Total area |
| 'finishedsquarefeet50' | Size of the finished living area on the first (entry) floor of the home |
| 'fips' | Federal Information Processing Standard code - see https://en.wikipedia.org/wiki/FIPS_county_code for more details |
| 'fireplacecnt' | Number of fireplaces in a home (if any) |
| 'fireplaceflag' | Is a fireplace present in this home |
| 'fullbathcnt' | Number of full bathrooms (sink, shower + bathtub, and toilet) present in home |
| 'garagecarcnt' | Total number of garages on the lot including an attached garage |
| 'garagetotalsqft' | Total number of square feet of all garages on lot including an attached garage |
| 'hashottuborspa' | Does the home have a hot tub or spa |
| 'heatingorsystemtypeid' | Type of home heating system |
| 'latitude' | Latitude of the middle of the parcel multiplied by 10e6 |

| | |
|------------------------------|--|
| 'longitude' | Longitude of the middle of the parcel multiplied by 10e6 |
| 'lotsizesquarefeet' | Area of the lot in square feet |
| 'numberofstories' | Number of stories or levels the home has |
| 'parcelid' | Unique identifier for parcels (lots) |
| 'poolcnt' | Number of pools on the lot (if any) |
| 'poolsizesum' | Total square footage of all pools on property |
| 'pooltypeid10' | Spa or Hot Tub |
| 'pooltypeid2' | Pool with Spa/Hot Tub |
| 'pooltypeid7' | Pool without hot tub |
| 'propertycountylandusecode' | County land use code i.e. it's zoning at the county level |
| 'propertylandusetypeid' | Type of land use the property is zoned for |
| 'propertyzoningdesc' | Description of the allowed land uses (zoning) for that property |
| 'rawcensustractandblock' | Census tract and block ID combined - also contains blockgroup assignment by extension |
| 'censustractandblock' | Census tract and block ID combined - also contains blockgroup assignment by extension |
| 'regionidcounty' | County in which the property is located |
| 'regionidcity' | City in which the property is located (if any) |
| 'regionidzip' | Zip code in which the property is located |
| 'regionidneighborhood' | Neighborhood in which the property is located |
| 'roomcnt' | Total number of rooms in the principal residence |
| 'storytypeid' | Type of floors in a multi-story house (i.e. basement and main level, split-level, attic, etc.). See tab for details. |
| 'typeconstructiontypeid' | What type of construction material was used to construct the home |
| 'unitcnt' | Number of units the structure is built into (i.e. 2 = duplex, 3 = triplex, etc...) |
| 'yardbuildingsqft17' | Patio in yard |
| 'yardbuildingsqft26' | Storage shed/building in yard |
| 'yearbuilt' | The Year the principal residence was built |
| 'taxvaluedollarcnt' | The total tax assessed value of the parcel |
| 'structuretaxvaluedollarcnt' | The assessed value of the built structure on the parcel |
| 'landtaxvaluedollarcnt' | The assessed value of the land area of the parcel |
| 'taxamount' | The total property tax assessed for that assessment year |
| 'assessmentyear' | The year of the property tax assessment |
| 'taxdelinquencyflag' | Property taxes for this parcel are past due as of 2015 |
| 'taxdelinquencyyear' | Year for which the unpaid property taxes were due |

The datasets have the same structure. The properties_2016 and properties_2017 .csv files have 58 features each and the train_2016 and train_2017 .csv files have 3 features each – parcelid, logerror and transaction time. Since logerror is the target variable for our model, we combine the relevant 2016 and 2017 files together to get a full dataset with features and target together. This join is on the parcelid. The output from Jupyter notebook gives the following result:

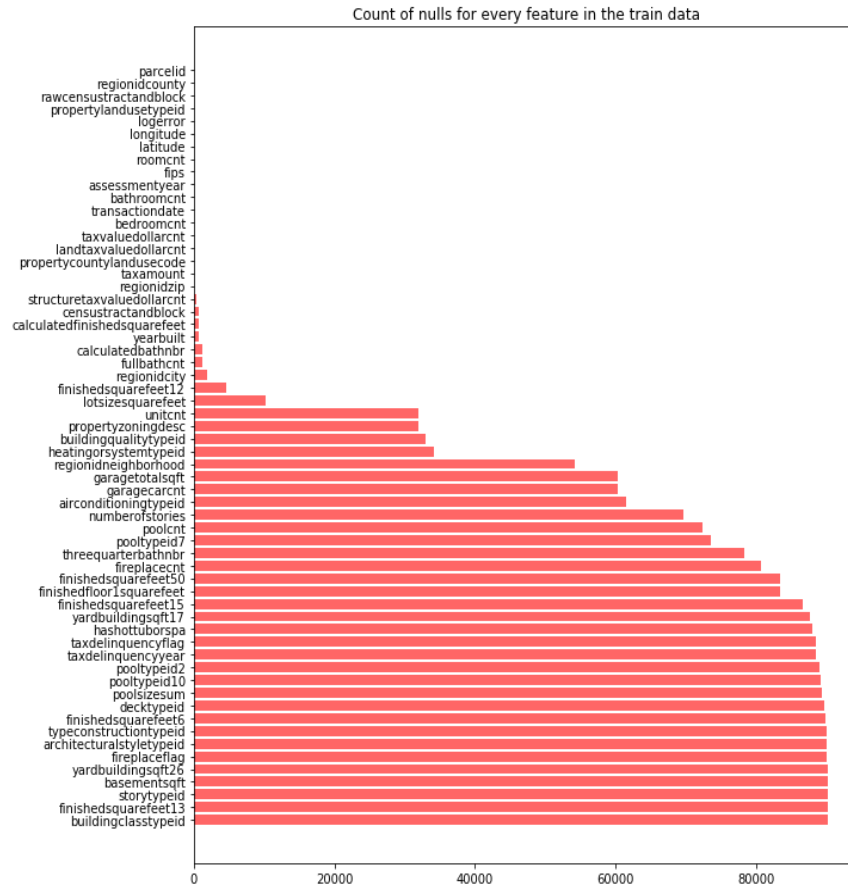
The descriptive statistics of the train dataset is analyzed and found that there are a lot of missing values in the dataset. The missing values needs to be addressed to proceed with the modeling. In this case, we resort to imputing with the feature

means to address missing data. But before that we study the features in more detail to identify redundant or highly correlated variables.

The count and extend of missing data can be understood from the following:

| feature index | features | nullcount |
|---------------|--------------------------|-----------|
| 8 | buildingclasstypeid | 90259 |
| 15 | finishedsquarefeet13 | 90242 |
| 43 | storytypeid | 90232 |
| 5 | basementsqft | 90232 |
| 48 | yardbuildingsqft26 | 90180 |
| 51 | fireplaceflag | 90053 |
| 4 | architecturalstyletypeid | 90014 |
| 45 | typeconstructiontypeid | 89976 |
| 18 | finishedsquarefeet6 | 89854 |
| 11 | decktypeid | 89617 |
| 30 | poolsizeum | 89306 |
| 31 | pooltypeid10 | 89114 |
| 32 | pooltypeid2 | 89071 |
| 58 | taxdelinquencyyear | 88492 |
| 57 | taxdelinquencyflag | 88492 |
| 24 | hashottuborspa | 87910 |
| 47 | yardbuildingsqft17 | 87629 |
| 16 | finishedsquarefeet15 | 86711 |
| 12 | finishedfloor1squarefeet | 83419 |
| 17 | finishedsquarefeet50 | 83419 |
| 20 | fireplacecnt | 80668 |
| 44 | threequarterbathnbr | 78266 |
| 33 | pooltypeid7 | 73578 |
| 29 | poolcnt | 72374 |
| 50 | numberofstories | 69705 |
| 3 | airconditioningtypeid | 61494 |
| 22 | garagecarcnt | 60338 |
| 23 | garagetotalsqft | 60338 |
| 40 | regionidneighborhood | 54263 |

| | | |
|----|------------------------------|-------|
| 25 | heatingorsystemtypeid | 34195 |
| 9 | buildingqualitytypeid | 32911 |
| 36 | propertyzoningdesc | 31962 |
| 46 | unitcnt | 31922 |
| 28 | lotsizesquarefeet | 10150 |
| 14 | finishedsquarefeet12 | 4679 |
| 38 | regionidcity | 1803 |
| 21 | fullbathcnt | 1182 |
| 10 | calculatedbathnbr | 1182 |
| 49 | yearbuilt | 756 |
| 13 | calculatedfinishedsquarefeet | 661 |
| 59 | censustractandblock | 605 |
| 52 | structuretaxvaluedollarcnt | 380 |
| 41 | regionidzip | 35 |
| 56 | taxamount | 6 |
| 34 | propertycountylandusecode | 1 |
| 55 | landtaxvaluedollarcnt | 1 |
| 53 | taxvaluedollarcnt | 1 |
| 7 | bedroomcnt | 0 |
| 2 | transactiondate | 0 |
| 6 | bathroomcnt | 0 |
| 54 | assessmentyear | 0 |
| 19 | fips | 0 |
| 42 | roomcnt | 0 |
| 26 | latitude | 0 |
| 27 | longitude | 0 |
| 1 | logerror | 0 |
| 35 | propertylandusetypeid | 0 |
| 37 | rawcensustractandblock | 0 |
| 39 | regionidcounty | 0 |
| 0 | parcelid | 0 |



The red color in the above figure indicates the extend of missing values. It can be seen that some features have majority of the features missing.

B. Exploratory Visualization:

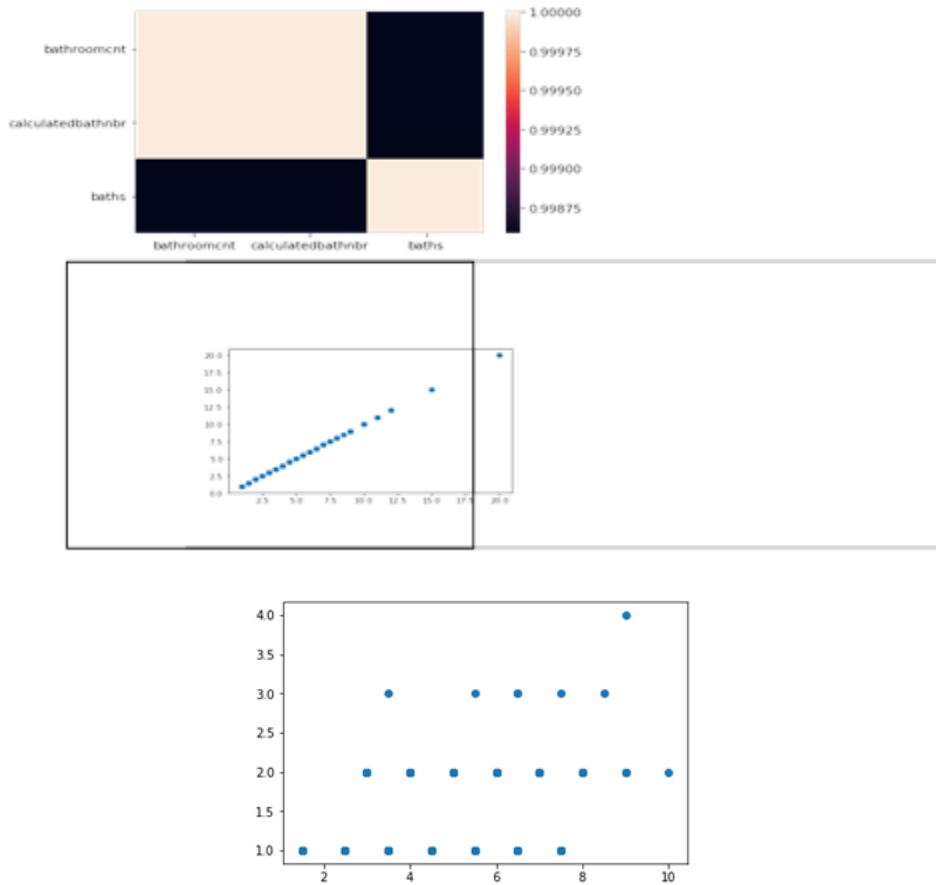
After exploring the data, it is seen that many variables represent the same or similar features in the data. Such duplicate features are identified and removed to prevent overfitting and multicollinearity. Overfitting and Multicollinearity are two common problems that we face during statistical modeling. Overfitting refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. If there are too many variables in the data fully or partially describing the same feature of the data, then there is a high tendency for the model to be over fit on these variables. Hence it is required to remove these variables. Similarly, multicollinearity exists whenever two or more of the predictors in a regression model are moderately or highly correlated. Multicollinearity can be identified by Pearson correlation coefficient or scatter plots.

For example, consider the following variables, in the data -

Features - 4,8,10,21 : 'bathroomcnt', 'calculatedbathnbr', threequarterbathnbr, fullbathcnt: All these represent the number of bathrooms in the house - either full or fractional. As mentioned, 'bathroomcnt' and 'calculatedbathnbr' are the same, as can be seen from the scatterplot and correlation below.

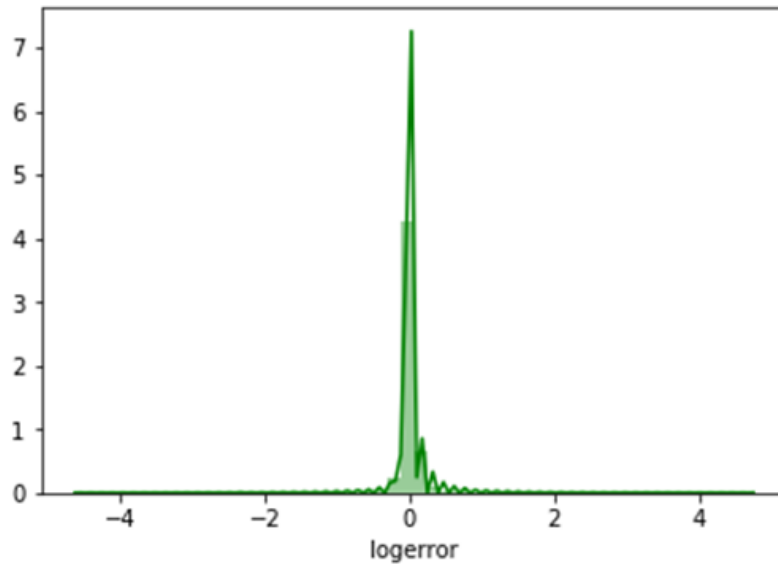
These are definitely one and the same. Bathroomcnt does not have any missing values as can be seen in the descriptive statistics and the missing data counts and hence only bathroomcnt is used for modeling and calculatedbathnbr is ignored. Again, plotting bathroomcnt against 'threequarterbathnbr', 'fullbathcnt' and the sum of 'threequarterbathnbr' and 'fullbathcnt' (a new variable called 'baths'), the following plots and correlations are obtained.

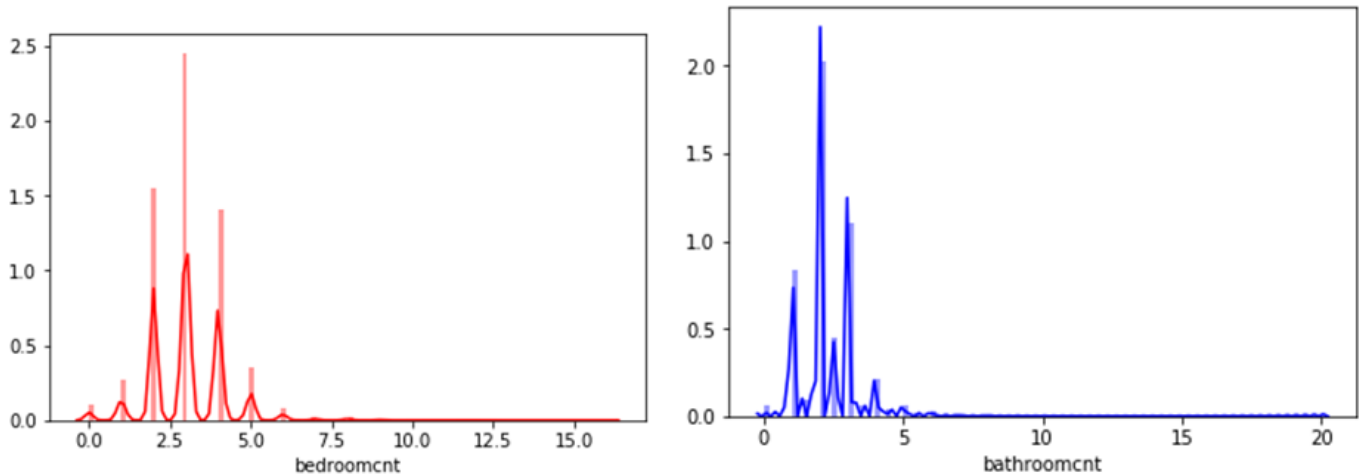
So bathroomcnt has a very high positive correlation between caculatedbathnbr and the sum of threequarterbathnbr' and fullbathcnt'. Hence bathroomcnt is used for modeling, as it explains well the other variables too.



The correlation between bathroomcnt and threequarterbathnbr is 0.260262959701

A histogram of the target variable – logerror and a few main features is drawn to check for normality. It is as shown below:





From the figure it is seen that the logerror variable is distributed normally. The bedroomcnt and bathroomcnt features are normal but a little skewed, which indicates the presence of some outliers.

C. Algorithms and Techniques:

After removing the duplicate features and addressing the missing values, we begin the modeling of the data. There are over 40 features even after duplicate variables are removed. But not all variables are equally important. Some techniques for feature selection are: Univariate selection, Recursive Feature Selection (RFE), Principal Component Analysis (PCA) and Feature Importance. Feature Importance is mostly available for classification models to understand the importance of each feature in the model. I have used regression techniques for modeling and have used RFE and PCA for analysis. It was found that PCA performs better for the same number of features.

A benchmark model is built on Linear Regression and further tuned with adding xgboost. Linear Regression is a classical regression model for predicting target variable based on the features given. xgboost is a special implementation of Gradient Boosting model which gives more efficiency and performance. I have selected these two models because Linear Regression is a very simple straight forward model and provides good results if implemented correctly. Xgboost on the other hand, is more complex in its implementation and was developed using key algorithm implementation features like sparse aware, block structure and continued training and provides better model execution speed and performance. xgboost was developed for more complex problems and is a winning algorithm for such problems. Selecting two algorithms at the opposite range of the spectrum helps to understand better, the model performance.

Multiple linear regression (MLR) is a method used to model the linear relationship between a dependent variable (target) and one or more independent variables (predictors).

$$\text{observed data} \rightarrow y = b_0 + b_1x_1 + b_2x_2 + \dots + b_px_p + \varepsilon$$

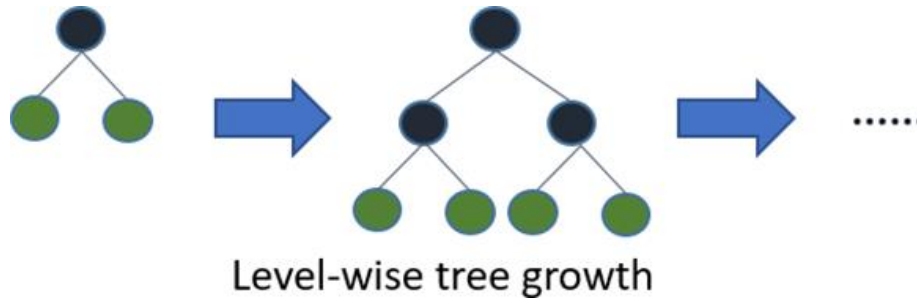
$$\text{predicted data} \rightarrow y' = b_0 + b_1x_1 + b_2x_2 + \dots + b_px_p$$

$$\text{error} \rightarrow \varepsilon = y - y'$$

The MAE that we attempt to find is the mean absolute value of all the ε for all records.

The MLR model is based on several assumptions (e.g., errors are normally distributed with zero mean and constant variance, that is homoscedasticity).

Xgboost implements gradient boosting model. Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. This approach supports both regression and classification predictive modeling problems. The Level-wise tree growth in XGBOOST is shown below.



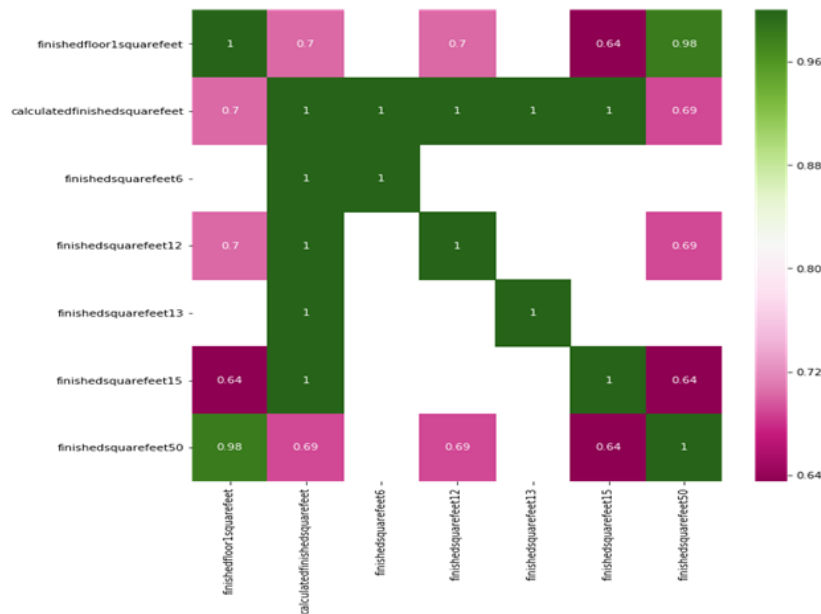
D. Benchmark:

My benchmark model is built on Linear Regression with and without Principal Component Analysis for 15 features. The model built with PCA offers a better result. This is because, though some variables are removed from the initial dataset, there are still extraneous variables which are not relevant to the target. PCA does a good job of picking the relevant features and giving a model based on these features. Principal Component Analysis (or PCA) uses linear algebra to transform the dataset into a compressed form. The dataset is split into train and validation sets on 80:20 ratio. Using PCA and Linear Regression, the MAE obtained is 0.0693

III. METHODOLOGY

A. Data Preprocessing

We have seen the analysis for the 'bathroomcnt' relative to other similar features. There are more features that I identified, which has high correlation with other variables. Some of them are: 'finishedfloor1squarefeet', 'calculatedfinishedsquarefeet', 'finishedsquarefeet6', 'finishedsquarefeet12', 'finishedsquarefeet13', 'finishedsquarefeet15' and 'finishedsquarefeet50'. It can be seen that 'calculatedfinishedsquarefeet' has high correlation with other variables. Hence it is sufficient to explain the variability of the other features too. We retain 'calculatedfinishedsquarefeet' in the model and omit the other features. The below heatmap explains, the high correlation between 'calculatedfinishedsquarefeet' and the other related features.



It is observed that correlation of 'calculatedfinishedsquarefeet' with the other 6 features is very high tending to 1.

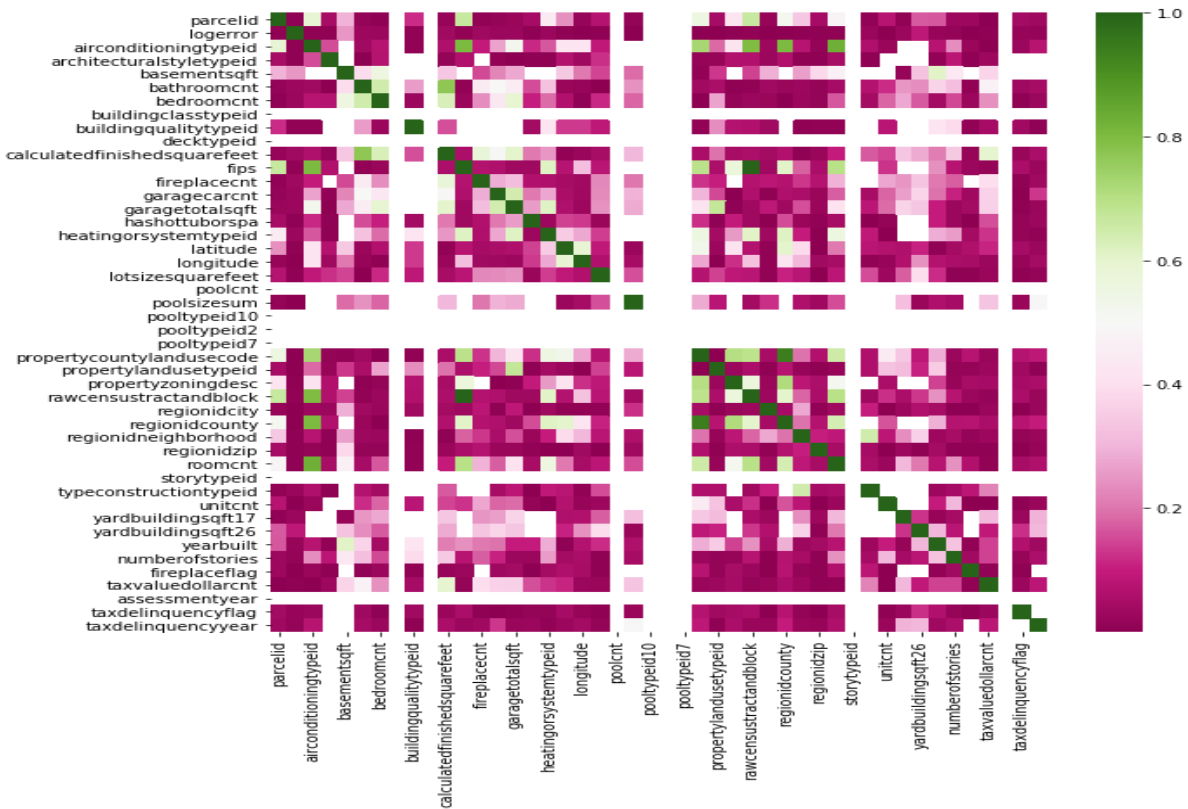
Another variable, which can be omitted is censustractandblock. 'censustractandblock' and 'rawcensustractandblock' are highly correlated with a value of ~1. Moreover, it can be seen from the data dictionary that they denote the same features.

Considering the features: 'taxvaluedollarcnt', 'structuretaxvaluedollarcnt', 'landtaxvaluedollarcnt', 'taxamount', we see similar high correlation.

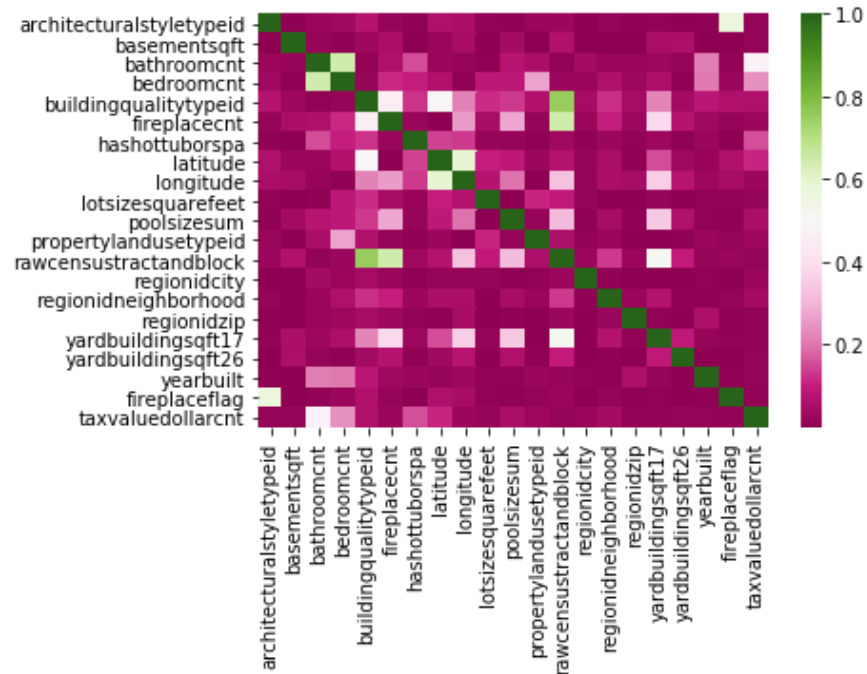


'taxvaluedollarcnt' is highly correlated with the other 3 features and hence retained in the model to explain the others.

After identifying all these variables, the unwanted features are removed. When we plot the heatmap again, there is significant improvement in the correlation. It is seen that there are only a few high correlations among unrelated features, which could be coincidental. The green color signifies a high correlation. As can be seen there are very few green pixels in the new heatmap.



For improving the features, I am doing a repeated pattern of identifying the high correlated variables with correlation values and heatmap and systematically removing them. In the end, I am using 21 variables which do not have a high correlation between them. The heatmap is as below:



There are a lot of missing data for a few of the features. Unless these values are imputed with suitable values, the model cannot be developed. There are different methods of missing data imputation.

- A constant value that has meaning within the domain, such as 0, distinct from all other values.
- A value from another randomly selected record.
- A mean, median or mode value for the column.
- A value estimated by another predictive model.

In this case, we decide to impute with the mean values of the features. The test data is also imputed with the mean values later as we did this to the train data for modeling.

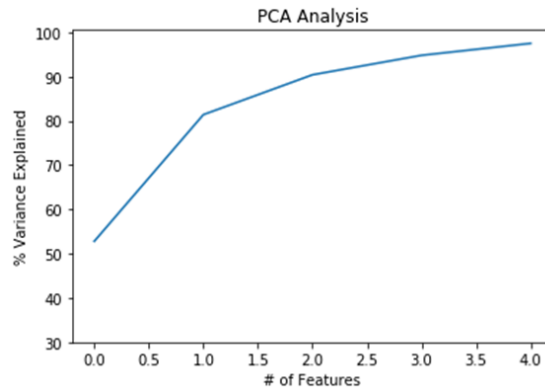
Another key step which needs to be done is converting the categorical features to numerical features. There are different ways of converting categories to numbers. OneHotEncoder and LabelEncoder are two methods available in sklearn package of Python. The drawback of LabelEncoder is that it taken ordinality into account, which means if 'A' is denoted by 1, 'B' by 2 and 'C' by 3, the LabelEncoder logic assumes that 'B' is the midpoint of 'A' and 'C'. Hence I resolved to use OneHotEncoding.

B. Implementation:

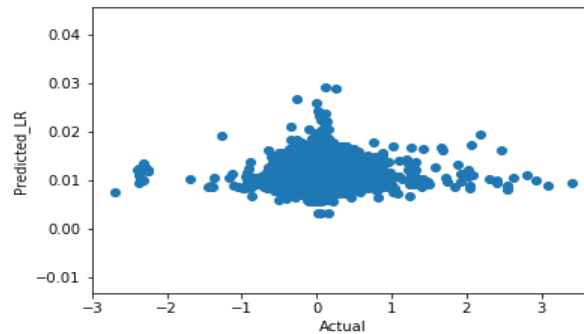
The training dataset is split into a training part and validation part. The benefit of this is that we train the model on the training part and test it on the validation part. This gives us an idea of the accuracy of the model, in case of a classification model or the (R-score) in case of a regression model. The train:valid split is at 75:25 ratio. After preprocessing the data, the final dataset has 90275 rows with 21 features. Splitting this into train:valid, we get training data as 67706 with 21 features and validation set is 22569 rows with 21 features.

PCA was employed for feature selection. PCA is used to find a small set of linear combinations of the covariates which are uncorrelated with each other. This will avoid the multicollinearity problem. It also ensured that the linear combinations chosen has maximal variance. A good regression design chooses values of the covariates which are spread out. The objective of PCA is to use only the first few components. I decided to use $n=5$ for the PCA components, which means the first 5 components will be

used. The results of the variance explained using these 5 components is : ([52.8, 81.4, 90.4, 94.8, 97.5]). This means the 1st component explains 52.8% of the variance, the 2nd component- 81.4% and finally the 5th component explains about 97.5% of the variance. Hence 5 components are a good selection. The plot of the variance explained for the number of components is shown below.



I studied the results of a Linear Regression model with and without PCA. The MAE obtained without PCA is 0.06677. A Pipeline is created to apply PCA and Linear Regression at the same time. This feature is available in the sklearn.pipeline package. The benefit of using pipeline is that it sequentially applies a list of transforms and finally fits the data to get a final estimator. The MAE obtained from Linear Regression with PCA is 0.06776. Without using PCA means the model is being built on the 21 features and with PCA, the model is built on the 5 best combinations of these 21 features. I decided to go with the Linear Model with PCA because it would perform better to avoid overfitting. The cross validation scores are obtained as [-0.00043186 -0.00105657 -0.00051924 -0.00087095 -0.00177083]. The scores are similar. The low values just indicate that there is not significant linear relation between features and target. A plot of the actual versus predicted target values is as shown for Linear Regression with PCA.



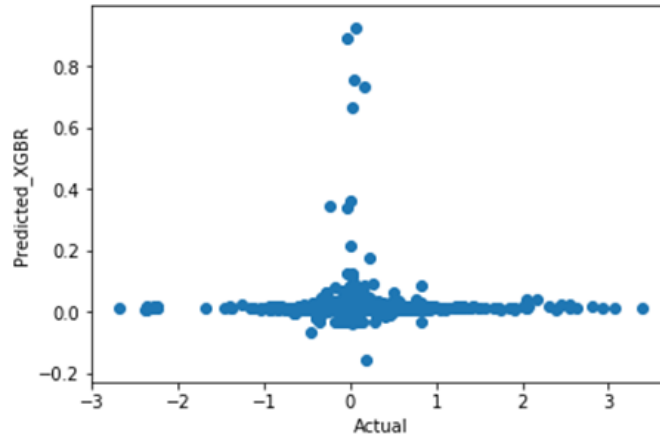
The next model developed implements xgboost, specifically an XGBRegressor, with and without PCA. The pipeline is used here to apply the PCA and xgboost transforms before fitting the data.

I studied the effect of various parameters in xgboost and selected the ones that gave best results. For the purpose of discussion, I have included two different parameters - the max_depth and learning_rate, to show their effect on the results.

| max_depth | learning_rate | Logerror for xgboost with PCA |
|-----------|---------------|-------------------------------|
| 3 | 0.1 | 0.6693 |
| 5 | 0.1 | 0.6724 |
| 3 | 0.08 | 0.6689 |
| 5 | 0.08 | 0.6702 |

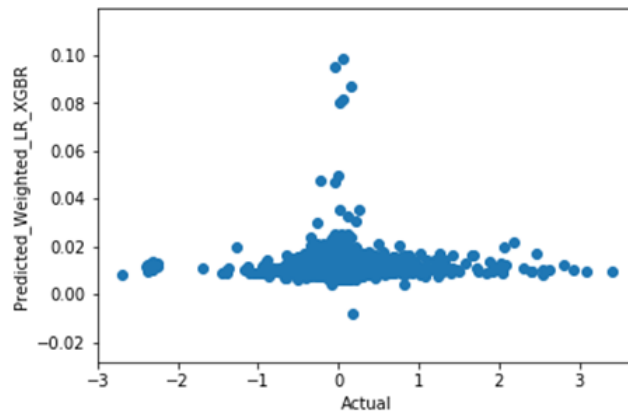
After comparing the various results from changing the parameters, I decide to go with max_depth = 3 and learning_rate =0.08

and default values for other parameters. Using these parameters, the logerror obtained for xgboost without PCA gives a MAE of 0.06666 and the one with PCA results in MAE of 0.06689



C. Refinement

In the third part, to improve the results, I applied a weighted model, which uses the best of Linear Regression with xgboost. The weightage used for xgboost is 0.2 and for LinearRegression, it is 0.8. A higher weight was chosen for the Linear Regression model with PCA since it performed better than the xgboost model with PCA. Applying this combined model, results in a MAE of 0.06675, which is a lesser value for MAE compared to all the other models studied. The plot of predicted versus actual is show below:



IV. RESULTS

A. Model Evaluation and Validation

The model results can be summarized as follow:

| Model | MAE Value |
|---|-----------|
| Linear Regression | 0.06677 |
| Linear Regression with PCA | 0.06676 |
| xgboost | 0.06666 |
| xgboost with PCA | 0.06689 |
| Weighted Model with Linear Regression and xgboost | 0.06675 |

B. Justification

From the different models, we studied the MAE could be reduced by a few decimal points, in the combined weighted

model.

The same preprocessing and transformations is done on the test data and the MAE is computed to be 0.0692. The test data comprises of only the middle 25000 records of the 2017 data.

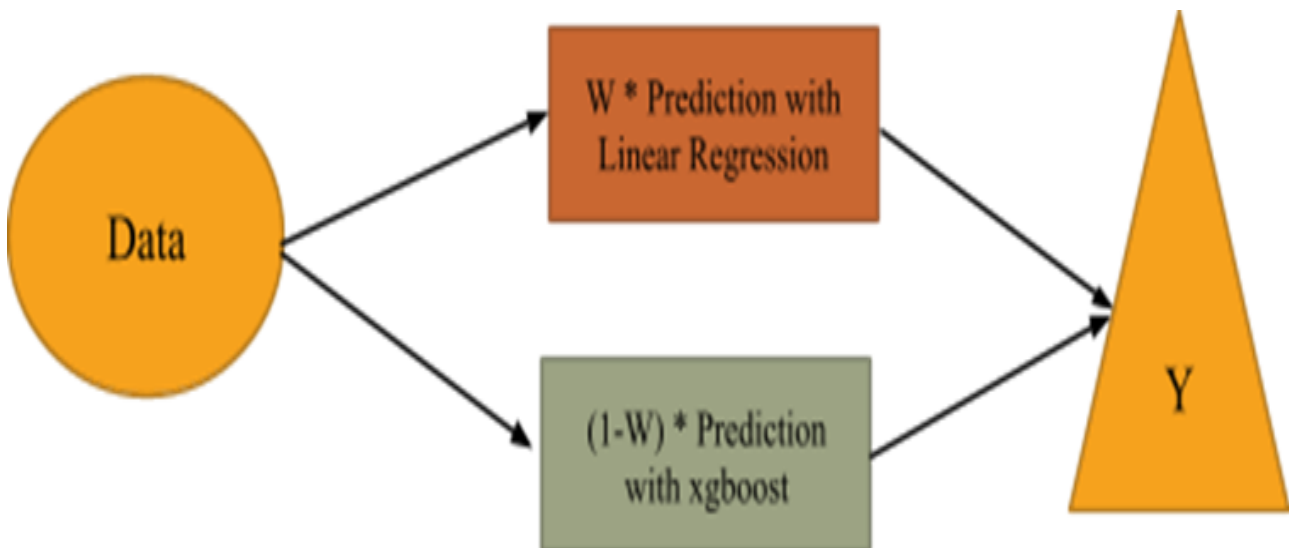
This result is reasonable because we have not considered the effect of socio economic factors in the model. House prices are also dependent on the year of sale and other environment factors, which have not been adequately added to the model. Again, we have not considered the role of perception of the buyer and the seller. There can be many factors that can lead to a higher or lower actual selling price of a house than the Zestimate. Higher prices can be attained, if a new commerce is coming up near the house to be sold. The market perception about the area will increase and result in higher sale price. Similarly, if a person is relocating and has to dispose of the house immediately, then he/she might settle for the best price, which might be lower than the actual value. There can be n number of scenarios similar to these, where the actual sale prices deviate from the Zestimate, which is outside the scope of the features that the dataset provides. Hence it is possible to get a different value than the values obtained while developing the model. But it can be seen there is not much difference and it can be further improved by studying the unknown factors and improving upon those.

V. CONCLUSION

The different steps I employed to develop the model are:

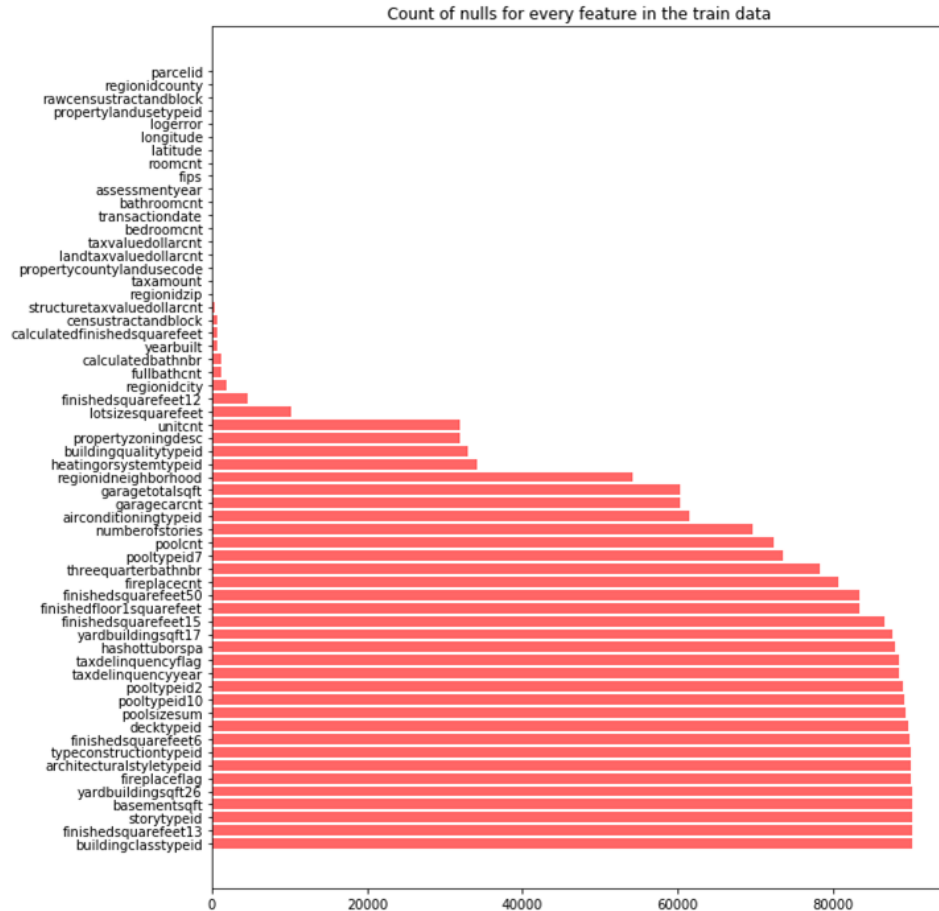
- Study the descriptive statistics of the given train data.
- Check missing or null values and outliers. Impute these values with a suitable number, which could be mean, median or any other.
- Identify high correlated values using scatterplots and heatmap.
- Use PCA for dimensionality reduction and feature selection.
- Developed a base model on Linear Regression Model.
- Build another model using XGBoost.
- Created a weighted model using Linear Regression and xgboost.
- Compared the MAE from different models and tested the model on the test dataset.

My final model can be visualized as follows:



A. Free-Form Visualization

Many visualizations were performed as part of the data analysis. One of the visualizations which contributed a lot to the analysis is the missing data graph. A data frame called nulls was created with counts of missing data for each feature. The counts were in ascending order. Hence when plotted, the features with the most missing data or nulls, appeared at the bottom of the graph. On the X-axis the counts are plotted and the Y-axis has the various feature names. This graph was very beneficial in the analysis, as it was aided in understanding the features with most null values, at a glance.



B. Reflection

The project was an implementation of the different techniques learnt during the coursework. Many concepts like descriptive statistics, plotting data to understand patterns, correlation analysis, missing value imputation, feature extraction and selection including PCA, regression analysis and boosting methods are implemented in this project. More than the final algorithms, the main thing that matters is the process that leads up to the model fitting. That is, even if very powerful algorithms are used, the prediction results depend on key elements like feature selection, how missing values is handled etc. Hence in my opinion, irrespective of the final algorithm, the fundamental steps like data exploration and feature selection needs to be given more importance and weightage in machine learning model development.

C. Improvement

Many improvements can be done on this model. We have not used grid search for the hyper parameter tuning. Again, different transaction periods have different number of sales activity. The primary reason for the varied activity is not explained in the dataset. More information of the reason for varied activity needs to be sought. For example, a company layoff can disrupt the housing industry in a specific region. But that is a temporary phase. But if there are major land calamities, then the effects of that will be spread over multiple years. It is difficult to understand from the dataset, if the trends in housing market is a temporary or long lasting one. Time series analysis can be employed for the effect of time on the variations, which we have not used here. A lot of methods have been employed comprehensively to understand the data. But there is still scope for more improvement and analysis based on the unknowns.

VI. REFERENCES

- [1] <https://onlinecourses.science.psu.edu/stat501/node/346> <https://www.kaggle.com/c/zillow-prize-1/data>
- [2] <https://web.stanford.edu/class/cs221/2017/restricted/p-final/ianjones/final.pdf> <https://machinelearningmastery.com/feature-selection-machine-learning-python> <https://machinelearningmastery.com>
- [3] <https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61be-ca4b6>
- [4] <http://www.stats.uwo.ca/faculty/braun/ss3850/notes/sas10.pdf> <https://seaborn.pydata.org/tutorial/distributions.html>

- [5] <https://plot.ly/matplotlib/histograms/>
- [6] <https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6>
- [7] <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgb-oost-with-codes-python/>