ZILLOW PRIZE
ZILLOW'S HOME VALUE PREDICTION (ZESTIMATE)

Authors:
Anuj Maheshwari, Probuddho Chakraboty, Laveena Kottwani

1 DATA EXPLORATION

Zillow is predicting the sale price of every house, which they call the 'Zestimate', comparing it with the actual sale price of the house and getting a log error of the difference. Given all the features of the house, we are going to predict this log error for other houses.

1.1 UNDERSTANDING THE DATASET

Data Description: Below is the description about the files used in this analysis/prediction:

- properties_2016_v2.csv - all the properties with their home features,has 2985217 observations with 53 variables for each observation
- train_2016_v2.csv - the training set with transactions, has 90275 observations with 3 variables for each observation

The training dataset contains the following 3 columns - parcelid, logerror, transactiondate.
We start with plotting a scatter plot and probability distribution to understand the distribution of 'logerror' in the train_2016 dataset.

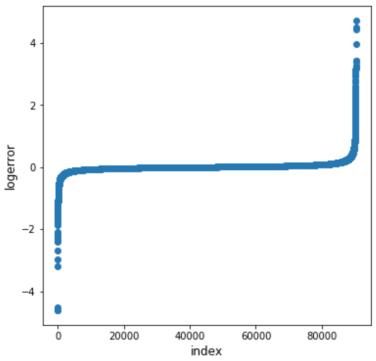

Figure 1: Scatter Plot

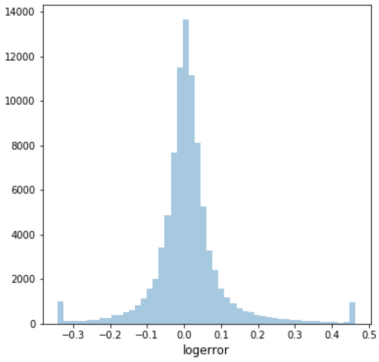

Figure 2: Distribution

We observe that most of the values are between 0.4 and -0.4, with some outliers at both ends, which we need to remove.

On exploring the transaction dates data, we see that the highest number of occurrences are in the month of June which means the highest number of properties were sold in June.

This also revealed that there was a significant drop in the number transactions during November and December 2016. Also, the data description mentioned that data for November and December 2016 were not complete.As we could see from the data page as well, the train data has all the transactions before October 15, 2016, plus some of the transactions after October 15, 2016. So we have shorter bars in the last three months.

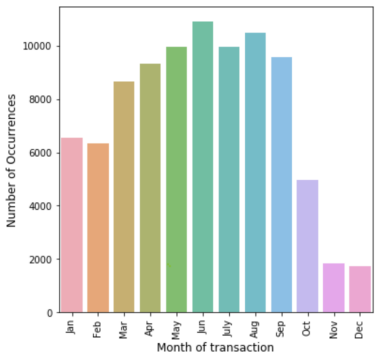

Figure 3: the number of transactions in each month

	column_name	missing_count	missing_ratio
6	basementsqft	90232	0.999524
9	buildingclasstypeid	90259	0.999823
16	finishedsquarefeet13	90242	0.999634
44	storytypeid	90232	0.999524

These 4 columns have missing ratio of greater than 99%!

Figure 4: Columns with highest missing values

2 Data Processing

We merge the data from train_2016_v2.csv and properties_2016.csv files on "parcelid" to facilitate EDA.

We observe that most of variables are of type float, hence we would convert all the variables to float type. We also bind float64 to float32 due to memory constraints.

Then we drop those features which are not needed during training. We drop 'parcelid', 'logerror', 'transactiondate','transaction_month','basementsqft', 'buildingclasstypeid', 'storytypeid', 'finishedsquarefeet13'. The columns basementsqft, buildingclasstypeid, finishedsquarefeet13 and storytypeid have missing ratio greater than 99percent hence we drop these features during training.

2.1 Filling missing values

We need to remove the NaN values in this dataset. Trying 3 different ways to substitute the NaN values- substituting NaN values with 0,substituting NaN values with -1 and substituting NaN values with mean of that.

Finally, we form our training and testing data by substituting NaN values with 0.

2.2 Feature Engineering

We select top features in our dataset using the following methods

2.2.1 XGBoost

Next we estimate the importance of features for a predictive modeling problem using the XGBoost library. We use xgboost to evaluate the importance of features. It provides with a score provides a value to help us estimate how useful it is to use in the construction of the boosted decision trees within the model. Using XGBoost, the important variables are 'latitude' followed by 'calculated finished square feet' and longitude.

2.2.2 Multicollinearity Analysis

Next we test for Multicollinearity with Variance Inflation Factors (VIF) score. We identify the correlation between independent variables and the strength of that correlation using the variance inflation factor (VIF).

we see low correlation of the target variables with all variables given. Having low correlation ensures among the features ensures that we do not see a lot of redundancy.

We observe that logerror reduce with increase in finishedsquarefeet12 ,calculatedfinishedsquarefeet and taxamount.

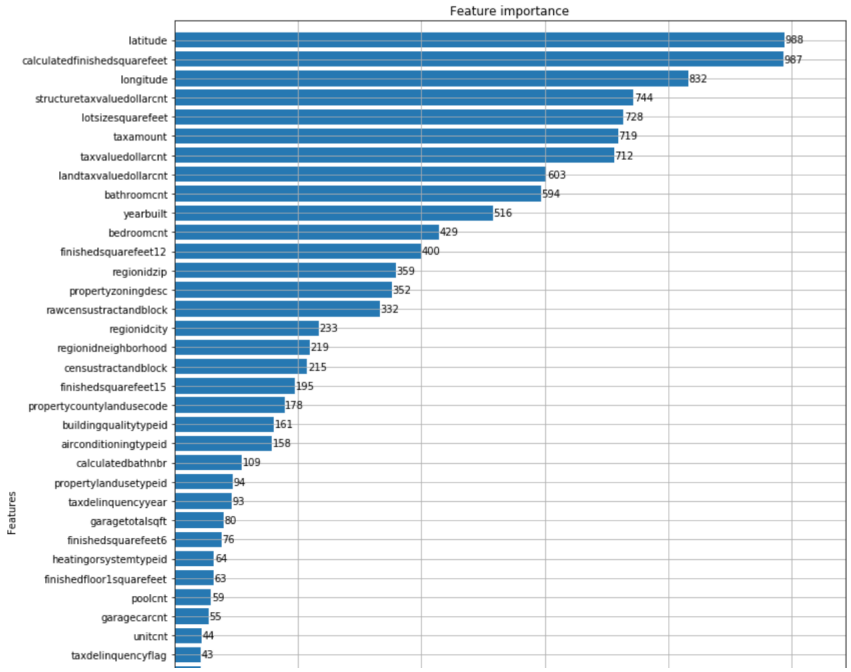

Figure 5: Feature Importance using XGBoost

There are few variables at the top of the graph without any correlation values. This could be that these values are singular and form no correlation with other variables.

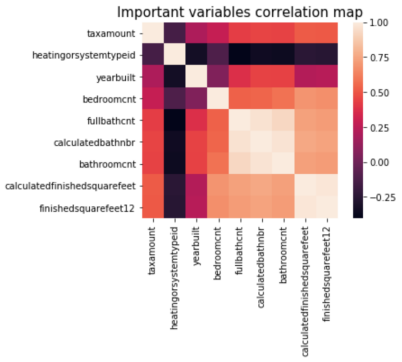

Figure 6: the number of transactions in each month

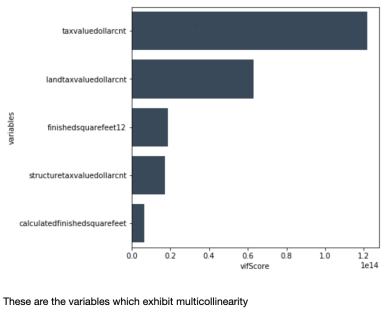

Figure 7: Columns with highest missing values

3 Model Exploration and Selection

3.1 Linear Regression

We implement a very simple linear regression model. This model is implemented using the linear model from sklearn library. Here, we substitute the NaN values with 0 instead of substituting NaN values with mean or -1.

3.1.1 Performance validation and Adapting to under- and over-fitting

We use Ordinary least squares Linear Regression model from the sklearn.linear_module linear regression library. We set n_jobs parameter to -1 to use all available CPUs and speedup the time for

evaluation of our large dataset.We observe the log error for the LR model to be 0.05476. Here we do see underfitting as the data is non-linear and not able to predict correct values.

3.2 XGBoost

We are implementing XGBoost Regression as well. Using xgboost library, we have set 'max_depth' to 4 and number of boosting rounds as 100. Here, we use the same data wherein we substitute the NaN values with 0 instead of substituting NaN values with mean or -1.

3.2.1 Performance validation and Adapting to under- and over-fitting

We have varied paramters - eta and max_depth to avoid overfitting and obtain the best possible values for them. Increasing max_depth and boost rounds avoids underfitting and shows low training error.

3.3 Neural Network

We are implementing a very simple neural network. We have defined a sequential model with 3 layers using ReLU and linear activation functions.Here, using the same data, we substitute the NaN values with 0 instead of substituting NaN values with mean or -1.

3.3.1 Performance validation and Adapting to under- and over-fitting

We define a simple neural network with 1 hidden layer, rectified linear unit as activation function and mean absolute error as metric. Since we are using very few layers, we find that the model is underfitting. Increasing the number of layers would help us get more accurate result. We have trained the model for a limited number of epochs, increasing the number of epochs would help to avoid under-fitting.

4 Conclusion

Below are the Mean Absolute Error for each of the models implemented.

Linear Regression	best Mean Abs. Error = 0.0547
XGBoost	best Mean Abs. Error = 0.0539
Neural Network	best Mean Abs. Error = 0.0553

References

https://www.kaggle.com/sudalairajkumar/simple-exploration-notebook-zillow-prize

https://www.kaggle.com/anokas/simple-xgboost-starter-0-0655