Springboard Data Science Career Track

Capstone Project 1: Finale Report
Zillow Home Valuation Prediction

Introduction

Buying a house is a massive investment and carry with it a great deal of risk. Having accurate models which forecast home values based on empirical data can help both home buyers and financial institutions make sound decisions.

I will analyse a dataset of real estate transactions provided by Zillow and use it to build and train a model which predicts the error of Zillows own prediction algorithm.

Data wrangling and cleaning

The dataset immediately provided two related problems. The first is that the dataset included entries which Zillow would use to evaluate the model. These values did not have a target or dependant variable associated with it and thus are useless to us. The other problem is that the dataset was quite large and unwieldy.

The solution to this was to use SQLalchemy to create a local database chunk by chunk and use SQL commands to filter and create a new csv with only those entries which have target values. The resulting csv was small enough to be easily handled by a personal computer.

Dealing with Missing Values

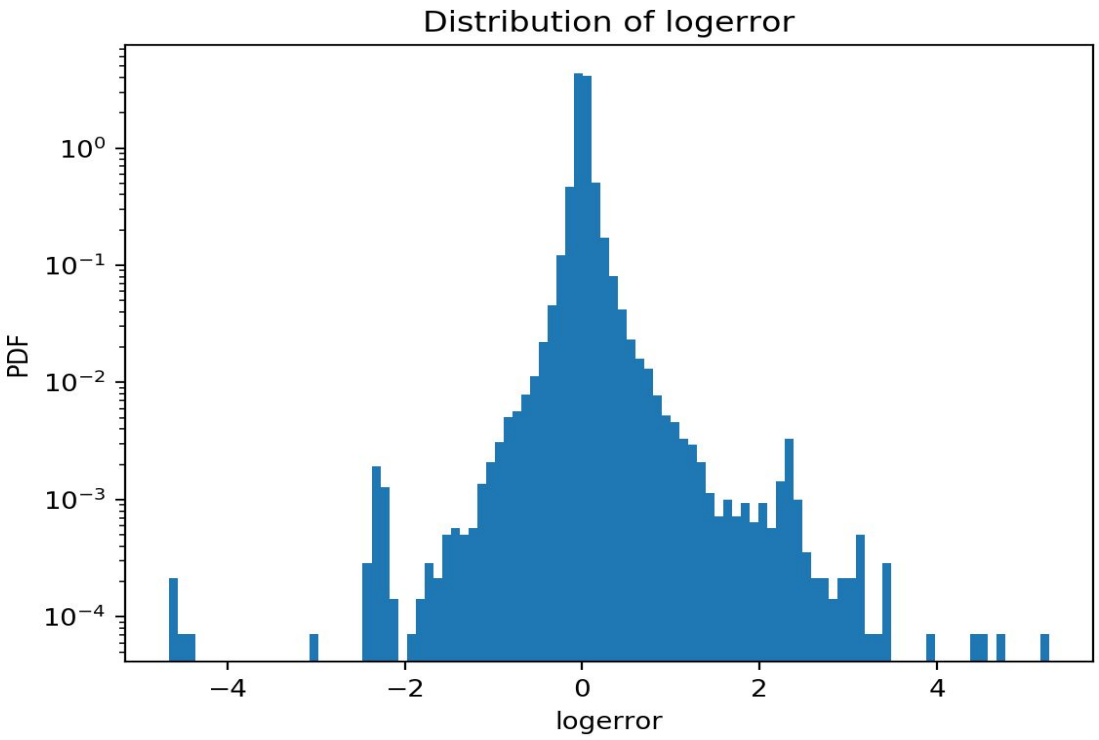
There were a number of missing values. Easier values to deal with were missing values represented False or 0, in which case these were filled in.

For some more difficult missing values I operated under the assumption that nearby properties share similar features. As such, I used K-nearest-neighbours with latitude and longitude to predict values for features like neighbourhood or floor count.

These processes were undertaken after some initial modelling performed poorly, so I needed to go back and do additional cleaning.

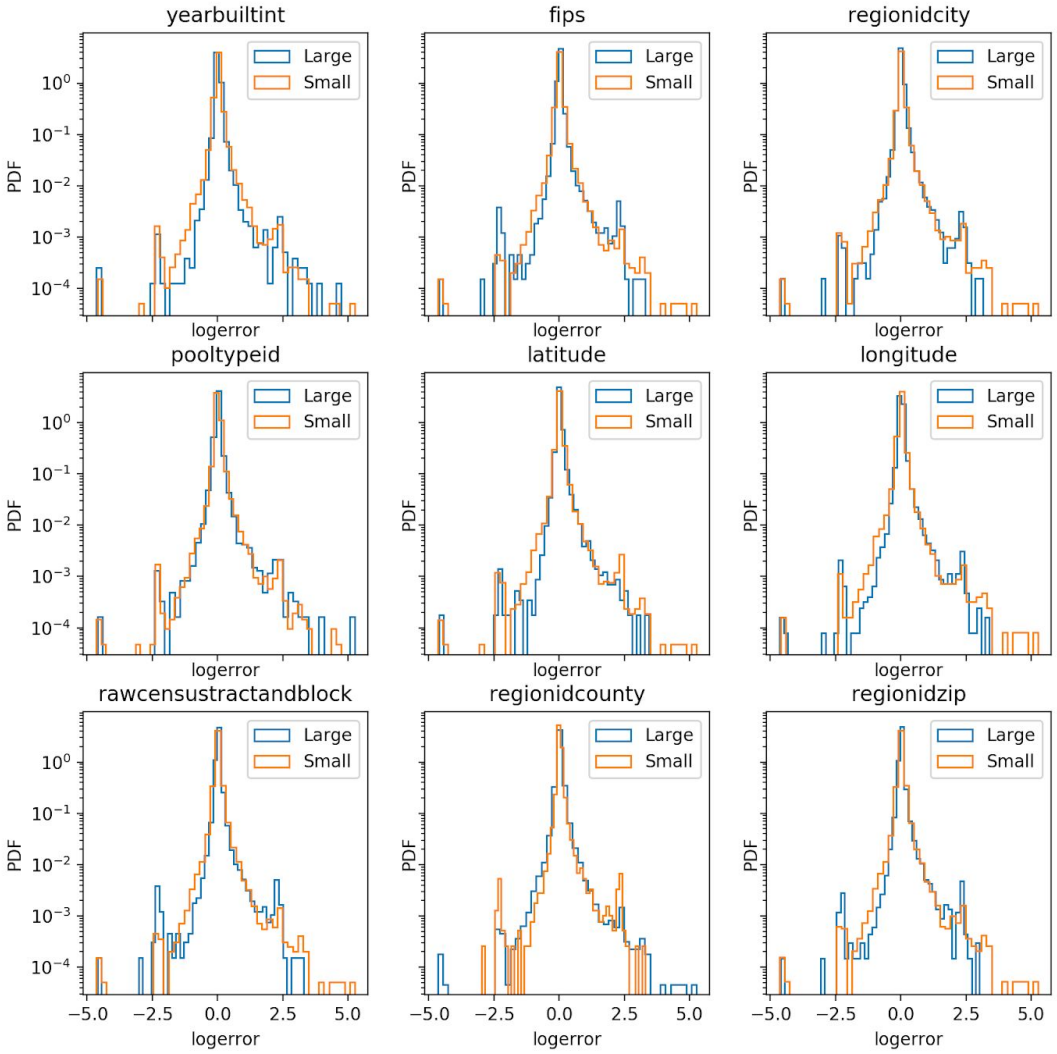
Exploratory Data Analysis

The next step was too look at different features and see if I could find any patterns. By looking at a histogram for the target variable 'logerror' I see that there are two clusters at around 2 and -2.



So I decided to look for features which may be have an outsized effect on the dataset causing these clusters. In doing so I identified 9 features which may have a different distribution for those values in the clusters. These were 'yearbuiltin,' 'fips,' 'regionidcity,' 'pooltypeid,' 'latitude,' 'longitude,' rawcensustractandblock,' 'regionidcounty' and 'regionidzip.' I then looked for a cutoff point in each feature which would separate the values into a group that included the clusters and a group that did not.

I then performed a two sample bootstrapping test on each feature to verify that there was a difference in the distribution between the portion of the entries which included the clusters in the logerror and the others which did not.



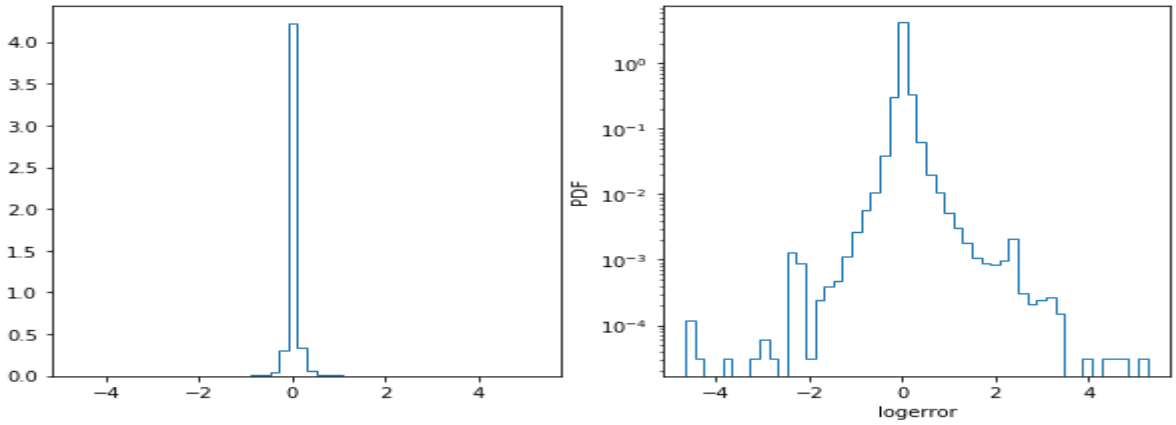
Nonetheless, I developed a more complete list by using gradient boosting from xgboost to find a list of important features as some of the above list turned out to be poor choices.

Initial Linear and Polynomial Regression

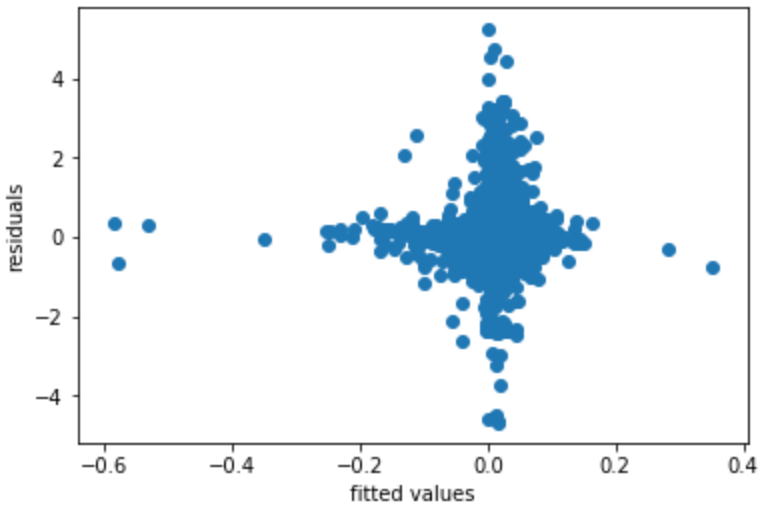
I started with a basic linear regression to get some initial information on how to fit a more in depth model. These initial attempts can be found in the notebooks Linear_Regression.ipynb and Linear_Regression_pt2.ipynb.

Initial Difficulties

The data for 'logerror' is nearly entirely closer to 0, thus linear regressions will be very close to the line $y = 0$. However, this means that I will miss many of the points which are further



from 0. I can see the distribution of 'logerror' in normal scale to show how prevalent 0 is, and also in log scale to the structure of the data not near 0.



A fitted linear regression on this data gives an r2 score of only 0.004 and a residuals plot, which should be randomly distributed around 0 is displayed below.

I did not expect a linear regression to be a very good model, but this is much worse than expected.

Even some basic polynomial regressions do not do a much better job with similar r2 scores.

Reactions

The major reaction to this was to go back to data cleaning and rework the missing data using K-nearest-neighbours to guess at missing values. Additionally, gradient boosting was used to determine which features were the most useful.

Ensemble Models

Using what was learned in earlier models, I decided to separate 'logerror' into bins and use a gradient boosting model as a classifier for on this set of 'logerror' bins. Then within each bin, use a ridge regressor. Unfortunately, this ensemble model was little better than a pure regressor since the the classifiers trained on this data saw too many entries in the bin containing 'logerror' = 0. This lead to the classifier placing everything from the test set into the 0 bin. The solution was to use sampling. Using SMOTETomek to undersample the bin containing 'logerror' = 0 and oversample the other bins.

This was done in the notebooks 'Classifier.ipynb' and 'Model.ipynb.' While this model performed better and better with each adjustment, it still performs somewhat poorly. In the above mentioned notebooks we observe that the classifier performs at 91% but the subsequent regressors perform quite poorly.

Dummy Model

We observed that the bulk of the data has a 'logerror' value of 0 and is also mostly symmetric about 0. Thus we can always use a dummy model that always predicts the 'logerror' as 0. In 'Model.ipynb' this model performs similarly to our ensemble model.

Conclusions

The nature of this data makes it extremely difficult to outperform the dummy model. Given the scope and timeframe of this project I was unable to find any deeper patterns in this data.

Thus my cursory conclusions about this data is that there is no substantial pattern that would aid the prediction tools at my disposal. Nonetheless, I was able to gain valuable experience using a number of skills I did not expect to use. Namely building an ensemble model, under and oversampling, and using SQLalchemy.