Udacity Machine Learning Engineer Nanodegree Capstone Project

Kaggle-Zillow Prize: Zillow's Home Value Prediction (Zestimate)

Allan Yong, 18/11/2017

Project Overview

This project is based on the Kaggle competition described at: Zillow's Home Value Prediction and seeks to improve the accuracy of house valuations. The median margin of error has improved over the years from 14% to 5%. However, reducing the error further has a meaningful impact on potential homeowners as a home is usually their largest purchase. Zillow has kindly agreed to provide access to a treasure trove of home valuation data in the hopes of obtaining a better valuation models for homes.

Kaggle's competition is appealing as a project as it allows the data scientist to explore various machine learning techniques without focusing on data collection.

Problem Statement

The inputs would be in the form of structured data where 2985217 data points and 59 features such as number of bedrooms will fed into a regressor-type machine learning algorithm (Sklearn's ExtraTreesRegressor, xgboost and lightbgm) to generate predictions of the difference between actual house price and Zillow's prediction.

Various machine learning techniques will be applied to the dataset to predict the mean absolute error (MAE) between the predicted log error and the actual log error of the house price. The simplest technique to approach this problem would be to apply multiple regression. This method seeks to establish a relationship between the 2 or more independent variables (house size, distance from center of the city) and a single dependent variable (mean absolute error of the log of house prices).

A collection of decision tree models in Xgboost or Lightgbm will use the features of the training data to make models to fit the training data. These models will then be saved and used to make a prediction on the test sets. The models are saved in practice to avoid having to re-run the preliminary models again when included in an ensemble of models.

Datasets and Inputs

Zillow has provided full list of real estate properties in three counties (Los Angeles, Orange and Ventura, California) data in 2016. There will be **2985217** data points with **58** features that will be used to predict the target variable (difference between actual house price and Zillow's prediction). Some of these features will be categorical such as airconditioningtypeid and numerical such as calculatedfinishedsquarefeet.

Data provided by Kaggle:

- **properties_2016.xlsx**
  All the properties with their home features for 2016
- **train_2016_v2.xlsx**

The training set with transactions from 1/1/2016 to 12/31/2016 with three columns: parceleid, logerror, and transactiondate

- **zillow_data_dictionary.xlsx**

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)
'buildingclasstypeid'	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 propert taxes were due

- **sample_submission.xlsx**
  A sample submission file in the correct format

Evaluation Metrics

The project success will be evaluated on the score improvement over the benchmark model, as given in the competition and the mean absolute error between the predicted log error and the actual log error of the house price. A lower error would be preferred.

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

The MAE will provide a simple measure of the prediction error that disregards the sign of the error and doesn't over-emphasize outliers.

$$MAE = \frac{\sum_{i=1}^{n} |y\_true_i - y\_pred_i|}{n}$$

The prediction time as well as training time will be recorded compared to estimate/quantify the computational workload required in a production environment. These times will be used with the final scores to determine viability of the model

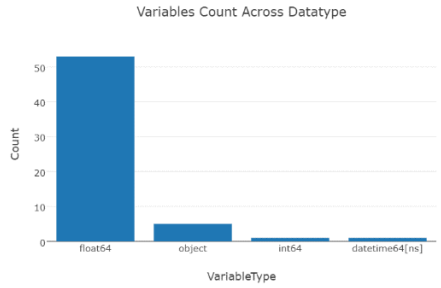
Benchmark

The benchmark model for this project has been supplied by Kaggle. However, the mean MAE could serve as a simple benchmark. Another benchmark could be a simple linear regression based on the actual data and the preprocessed data. The final model with be compared these simple models in order to judge its performance and quantify its improvements over the most simplistic model.

Analysis

1. **Data Exploration**

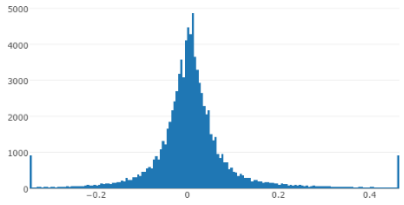
   Visualising datatypes:



While most of the features 54 are numerical in nature such as room count, there are 5 categorical features such as '*hashottuborspa', 'propertycountylandusecode', 'propertyzoningdesc', 'fireplaceflag', 'taxdelinquencyflag'*. The datetime object would be the '*transactiondate'*. The first 5 rows of the training data with columns truncated is shown below:

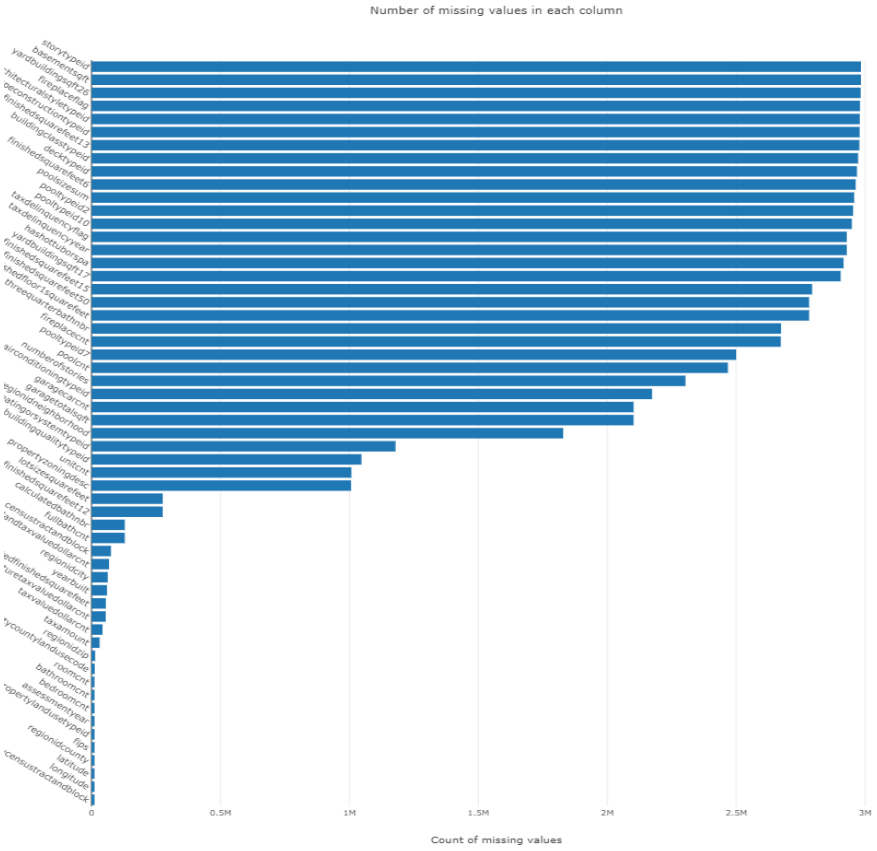
	parcelid	logerror	transactiondate	airconditioningtypeid	architecturalstyletypeid	basementsqft	bathroomcnt	bedroomcnt	buildingclasstype
0	11016594	0.0276	2016-01-01	1.0	NaN	NaN	2.0	3.0	NaN
1	14366692	-0.1684	2016-01-01	NaN	NaN	NaN	3.5	4.0	NaN
2	12098116	-0.0040	2016-01-01	1.0	NaN	NaN	3.0	2.0	NaN
3	12643413	0.0218	2016-01-02	1.0	NaN	NaN	2.0	2.0	NaN
4	14432541	-0.0050	2016-01-02	NaN	NaN	NaN	2.5	4.0	NaN

Target Variable Distribution



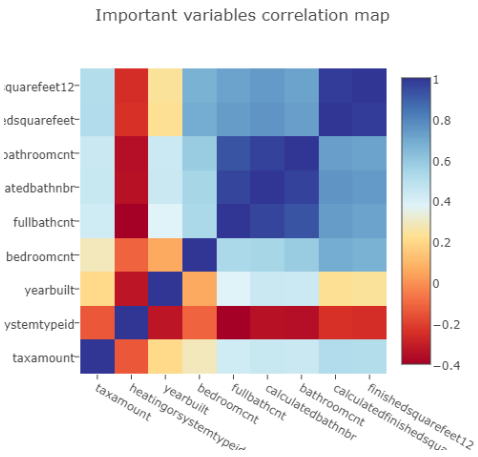
The target variable which is the logerror has a shape similar to a bell curve for the non-extreme portion of the graph. However, it is distorted by the outliers at both the far ends of the distribution.

Missing data



It is found that there are 9 features that are missing more than 99.5% of the data: *'architecturalstyletypeid', 'basementsqft', 'buildingclasstypeid', 'finishedsquarefeet13', 'finishedsquarefeet6', 'storytypeid', 'typeconstructiontypeid', 'yardbuildingsqft26'*, and *'fireplaceflag'.*

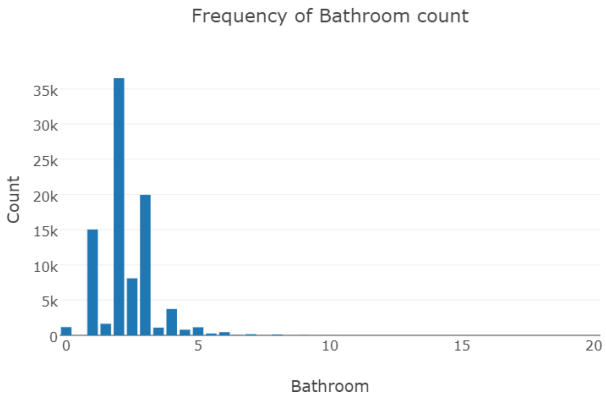
Correlation Analysis

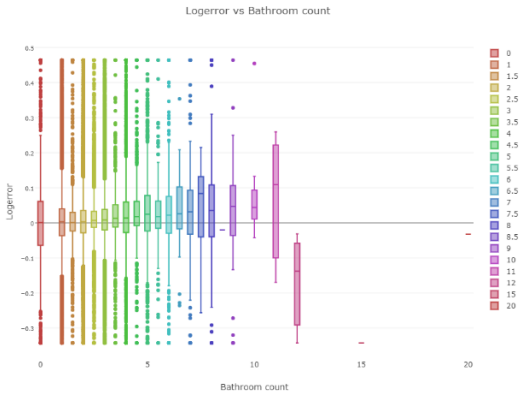


The correlation coefficient for all the features are calculated and the coefficients are found to be low overall. The feature with the largest coefficients are *'taxamount', 'heatingorsystemtypeid', 'yearbuilt', 'bedroomcnt', 'fullbathcnt', 'calculatedbathnbr', 'bathroomcnt', 'calculatedfinishedsquarefeet'*, and *'finishedsquarefeet12'.* A correlation heatmap of these variables are then produced. The heatmap confirms that *'calculatedfinishedsquarefeet'* and *'finishedsquarefeet12'* have a correlation of almost 1 and hence are practically redundant features. The same is true for the 3 features: *'fullbathcnt', 'calculatedbathnbr',* and *'bathroomcnt'.*

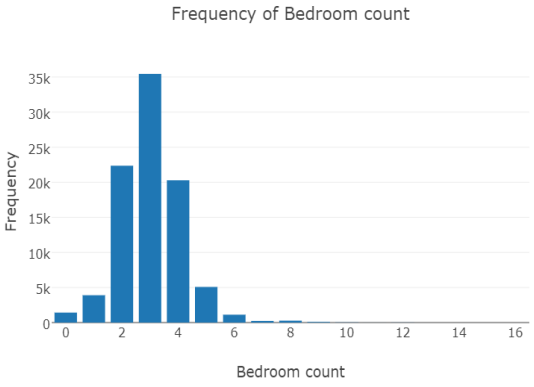
2. Exploratory Visualization

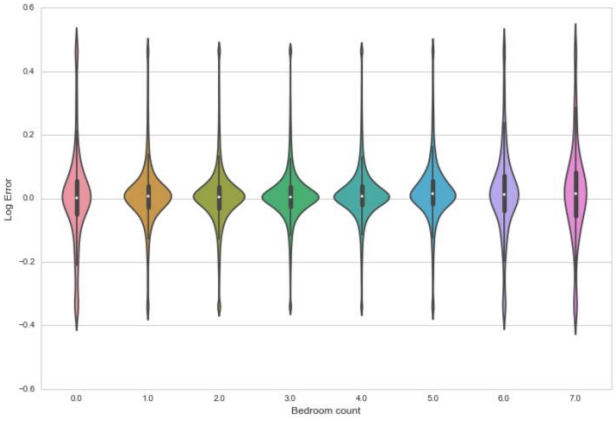
A few features discussed before such as *'bedroomcnt', 'bathroomcnt'* and *'taxamount'* will be investigated further.





Due to the scarcity of homes with 5 or more bathrooms and homes with no bathrooms, the logerror variance for these properties tend to be larger.





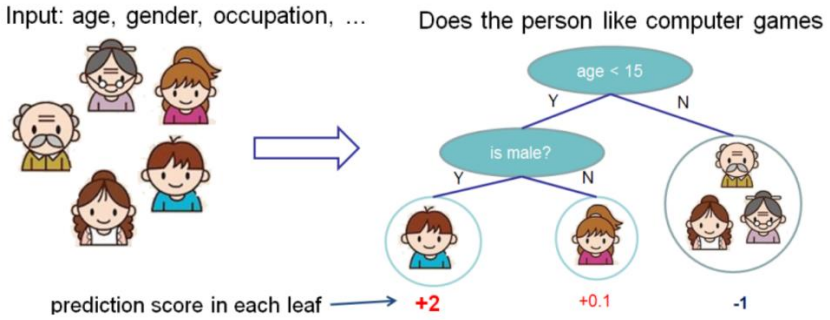
The larger logerror variance for properties with more than 5 bedrooms or zero bedrooms is again reflected in the violin plots for bedroom count.

3. Algorithms

Extra-trees regressor which is a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

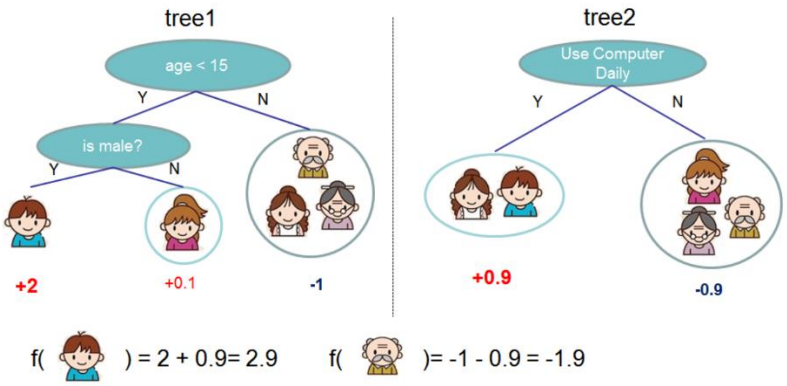
GradientBoostedRegression such as Xgboost and Lightgbm was applied to the problem due to their effectiveness and success at previous Kaggle competitions.

At the heart of the **Gradient Boosted Trees** is the tree ensemble model which is a set of classification and regression trees (CART). An example of a CART that classifies whether someone will like computer games.



Members of a family are classified into different leaves, and assigned a score. For decision trees, the leaf contains only decision values. For CART, a real score is associated with each leaf, allowing further optimization later on in the process of adding trees.

Since a single tree is not reliable in practice, a tree ensemble model, which sums the prediction of multiple trees together is used. An example of a tree ensemble of two trees.



The prediction scores of each individual tree are summed up to get the final score. The two trees try to *complement* each other.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

Where K is the number of trees, f is a function in the functional space $\mathcal{F}$, which is the set of all possible CARTs. Our objective to optimize can be written as

$$\text{obj}(\theta) = \sum_{i=1}^{n} L(\, y_i \,, \hat{y}_i) + \sum_{i=1}^{n} \Omega(\, f_k)$$

Where L is the training loss function, Ω is the regularization term and θ is the set of parameters to optimise.

Since it is difficult to train all the trees at once. An additive strategy is used. One new tree at a time to the existing ensemble. The prediction value at step t, $\hat{y}_i^{(t)}$

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

The objective function:

$$obj^t = \sum_{i=1}^{n} L(\, y_i \,, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant$$

With the Mean Squared Error (MSE) as the loss function:

$$obj^t = \sum_{i=1}^{n} (\, y_i - (\hat{y}_i^{(t-1)} + f_t(x_i))\,)^2 + \Omega(f_t) + constant$$

$$= \sum_{i=1}^{n} [2(\hat{y}_i^{(t-1)} - y_i)\, f_t(x_i) \; + f_t(x_i)^2\,] + \Omega(f_t) + constant$$

Comparing with Taylor expansion of loss function:

$$obj^t = \sum_{i=1}^{n} [L(y_i \,, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) \; + \frac{1}{2} h_i f_t^2\,(x_i)\,] + \Omega(f_t) + constant$$

Where

$$g_i = \partial_{\hat{y}_i^{(t-1)}} L(y_i \,, \hat{y}_i^{(t-1)},\; h_i = \partial^2_{\hat{y}_i^{(t-1)}} L(y_i \,, \hat{y}_i^{(t-1)}$$

Removing constants:

$$obj^t = \sum_{i=1}^{n} [\, g_i f_t(x_i) \; + \frac{1}{2} h_i f_t^2\,(x_i)\,] + \Omega(f_t)$$

Note that the optimization goal for the new tree only depends on g_i and h_i. Other custom loss functions are easily accommodated just by changing g_i and h_i.

A refined definition of a tree, $f(x)$:

$$f_t(x) = w_{q(x)} \,, w \in R^T, \qquad q\colon R^d \to \{1,2,\dots,T\}$$

w is the vector of scores on leaves, q is a function assigning each data point to the corresponding leaf, and T is the number of leaves.

Xgboost definition of complexity:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Objective function:

$$obj^{(t)} \approx \sum_{i=1}^{n} [\, g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \,] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \,] + \gamma T$$

Where $I_j \rightarrow \{i | q(x_i) = j\}$ is the set of indices of data points assigned to the j-th leaf.

Substituting $G_j = \sum_{i \in I_j} g_i$, $H_j = \sum_{i \in I_j} h_i$,

Objective function:

$$obj^{(t)} = \sum_{j=1}^{T} [(G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \,] + \gamma T$$

Differentiating with respect to w_j, $w_j^* = -\frac{G_j}{H_j + \lambda}$

Objective function:

$$obj^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

obj^* measures the fitness of a tree structure $q(x)$.

Other gradient boosting libraries such as Xgboost, Lightgbm and Catboost operate on the same principle but have different implementation in the tree boosting training phase and possibly their definition of complexity.

The reason these gradient boosting algorithms are selected for this task is due to their performance in terms of producing superior results, flexibility in dealing with data, and computational efficiency

Methodology

1. Data Preprocessing and feature engineering

Categorical features such as '*hashottuborspa', 'propertycountylandusecode', 'propertyzoningdesc', 'fireplaceflag', 'taxdelinquencyflag'* are removed before using then the training data is fed into Sklearn's Ensemble ExtraTreesRegressor algorithm. The categorical features are removed as this improved the results slightly. This may be due to the large portion of missing data for these features.

In addition, the Nan values are replaced by the mean values for the algorithms to work properly on the data.

2. **Implementation**

Parameters for **ExtraTreesRegressor:**

- **n_estimators**=100, # number of trees in the forest
- **max_depth**=30, # maximum depth of the tree
- **max_features**=0.3, # number of features to consider when looking for the best split
- **n_jobs**=-1, #the number of jobs is set to the number of cores.
- **random_state**=777

Parameters for **xgboost:**

- 'eta' : 0.02,
- 'objective' : 'reg:linear',
- 'eval_metric' : 'mae',
- 'max_depth' : 4,
- 'silent' : 1,

Parameters for **lightgbm:**

- 'max_bin' : 20, # maximum bin size
- 'learning_rate' : 0.0021, # shrinkage_rate
- 'boosting_type' : 'gbdt',
- 'objective' : 'regression',
- 'metric' : 'l1', # or 'mae'
- 'sub_feature' : 0.5, # feature fraction
- 'bagging_fraction': 0.85, # sub_row
- 'bagging_freq' : 40, # bagging frequency
- 'num_leaves' : 512, # number of leaf
- 'min_data' : 500, # minimum data in leaf
- 'min_hessian' : 0.05, # minimum sum of hessian in leaf

Parameters are obtained from some of the Kaggle scripts to avoid the time consuming hyperparameter phase.

3. **Refinement**

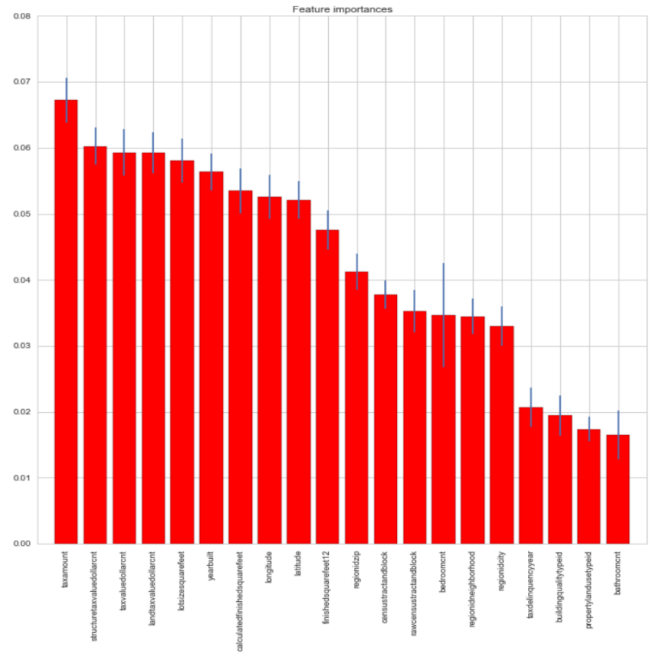
In order to improve the performance of Sklearn's ExtraTrees Regressor, some of its hyper parameters 'max_depth' , 'max_features', 'n_estimators'. Their optimised values are 50, 0.5, and 50 respectively., but this leads to ove rfitting as the MAE increased to 0.089835 compared to 0.076131 when the values are 50, 0.3 and 30 respecti vely. Further finetuning of the parameters are not carried out due to the discouraging results and the long tu ning time involved.

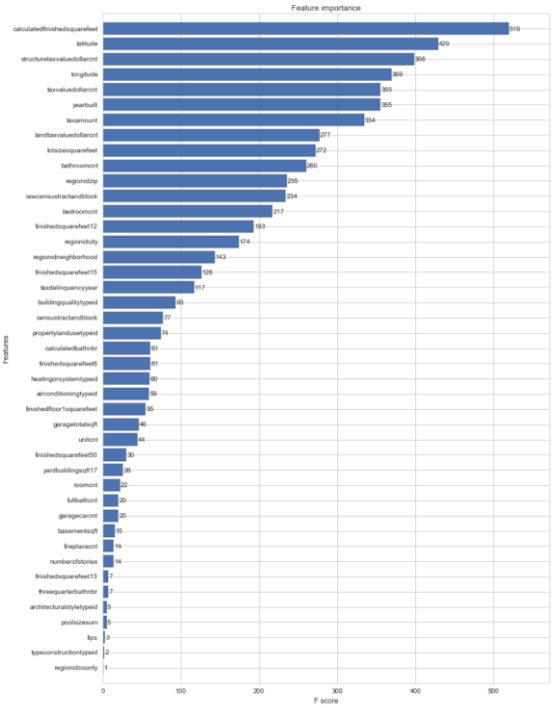
Another refinement that could be implemented would be to only use the top 10 features in descending order of importance or only features which are most independent from each other. Independent features will have low correlation coefficients.

However, removing some of the features seem to negatively impact the performance of Sklearn's Linear Regression and ExtreTreesRegressor. For Linear Regression, the MAE worsened from 0.060771 to 0.067238 when some features ('transactiondate') are removed. For ExtreTreesRegressor, the MAE worsened from 0.061565 to when so 0.07613.

Feature Importance

Sklearn's Ensemble ExtraTreesRegressor is used to find the order of importance of the features.





The **Extra-Tree Regressor** suggest that the top 10 most important features are the taxamount, structuretaxvaluedollarcnt, taxvaluedollarcnt, landtaxvaluedollarcnt, lotsizesquarefeet, yearbuilt, calculatedfinishedsquarefeet, longitude, latitude and finishedsquarefeet12 while **xgboost**'s top 10 features are calculatedfinishedsquarefeet, latitude, structuretaxvaluedollarcnt , longitude, taxvaluedollarcnt, yearbuilt, taxamount, landtaxvaluedollarcnt, lotsizesquarefeet and bathroomcnt.

There are 8 features that appear in the top 10 features in both list confirm that the **size, location, age** and **tax** associated with the property are important features.

When only the top 10 features from xgboost are used on lightgbm, the MAE worsens slightly from 0.0661294 to 0.066204. This means that the other features still contain extra information not contained or paritally independent of the top ten features.

Results

1. **Model Evaluation and Validation**

   The data is roughly split where 90% of the training data is used to train and the remaining 10% is used for the validation phase.

   A benchmark of just using the mean logerror as a prediction of the logerror yields an MAE of 0.112239 which is far higher than that given by the regression models. This shows that the regression models provide value and produce a superior result compared to the most basic mean prediction.

   Another basic benchmark would be to use the linear regression model provided by Sklearn. The MAE for the linear regression came out to be 0.067238. which is better than the ExtraTreesRegrssor even after some hyperparameter tuning.

Mean Absolute Error	
ExtraTreesRegressor (validation set)	0.075430
Linear Regression	0.067238
Xgboost (validation set)	0.066835
Lightgbm (validation set)	0.066116
Lightgbm (test set)	0.064662

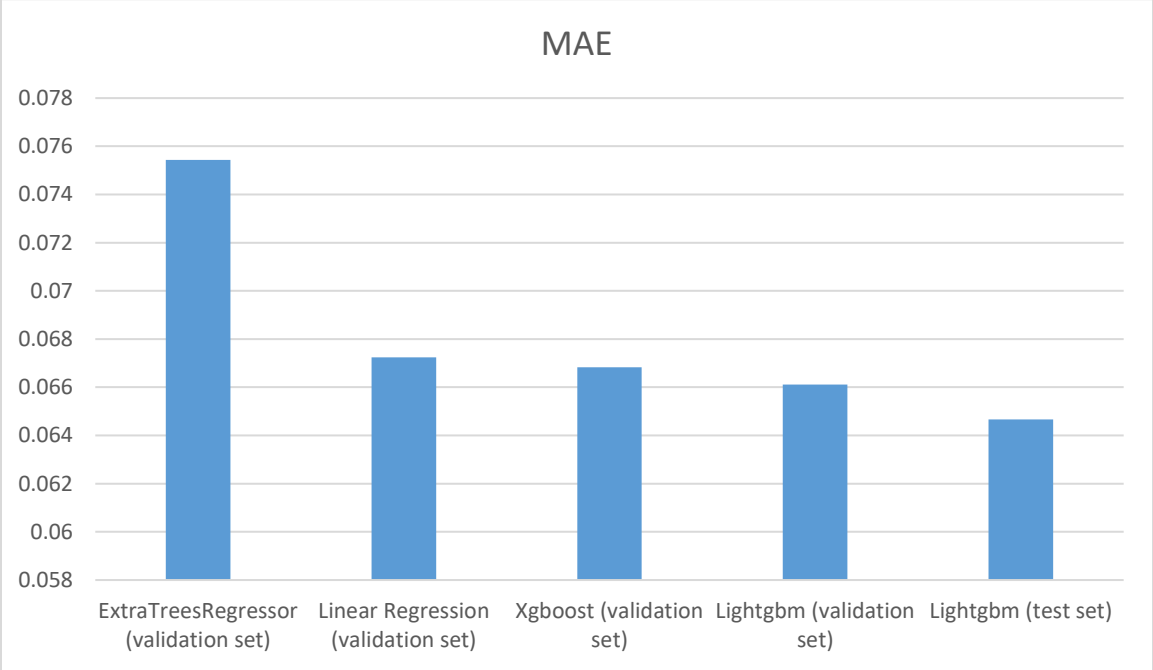
   Lightgbm performed the most favourably in the validation set and has the lowest MAE of 0.0646619 on the test set, showing that it is superior and less prone to overfitting compared to the other tree based gradient boosted regressors.

2. **Justification**

   The best MAE achieved using Lightgbm on the test set is 0.0646619 which is only marginally worse (2.3% higher) than the best score on the current leader board at 0.0631885, showing that the MAE is very difficult to improve on.

Conclusion

1. Free Form visualization



The MAE of the Lightgbm on the validation set is clearly superior to that of the other regression models. It is also encouraging to that the MAE on the test set is slightly lower in the test set. This suggests that the Lightgbm algorithm did not overfit the data despite the lower validation set MAE.

2. Reflection

This Kaggle competition and capstone project has taught me how to question the data, develop meaningful visualisations (utilise plotly, seaborn, ggplot), deal with missing data and imputations (sklearn), feature engineering (sklearn, xgboost) and utilise other python packages such as xgboost and lightgbm. This project has also given me the confidence to try out new packages such as Catboost in the future which has an inbuilt and optimised one-hot encoder to deal with categorical data without much preprocessing.

3. Improvement

The results may be improved via an ensemble of sklearn's regressors, xgboost and lightgbm to build a more robust model that does not overfit. Extra feature engineering where new features are developed and/or features are clustered using unsupervised learning may also help improve the MAE score slightly. More parameters for the algorithms could be explored and finetuned should more computing power and time be made available. Other algorithms such as Catboost may be explored given its superior performance over xgboost and lightgbm on some datasets.