Zestimate - Kaggle

Capstone #1 - Theo Pujianto

Problem Statement

Real estate is one of the largest business sectors in the U.S. There are new sellers and new buyers emerging every day. The major challenge is to put the price tag on the house.

One of the largest online real estate platform is Zillow. Zillow provided house price prediction as one of its staple features.

The objective of this project is to predict the logerror between the predicted log error and the actual log error. The log error is defined as:
logerror=log(Zestimate)−log(SalePrice)
In other words, the goal of the project is to predict the difference between the Zestimate and the actual sales price of homes.

Client

Zillow (obviously), real estate sellers, buyers, or even agents will benefit from this project.

Data

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

Kaggle provided real estate data from three counties in Southern California: Los Angeles, Ventura County, and Orange County. There are (2) sets of data given: training set with the actual logerror and house features data. The house features data consisting of 2,985,217 observations (rows), each with 56 features. The training set, however, has 90,275 observations (rows).

Approach

Linear regression machine learning models were utilized for this project. Features from the data were used as the predictors to train the model.

Pycaret module was used as the main machine learning module.

Workflow

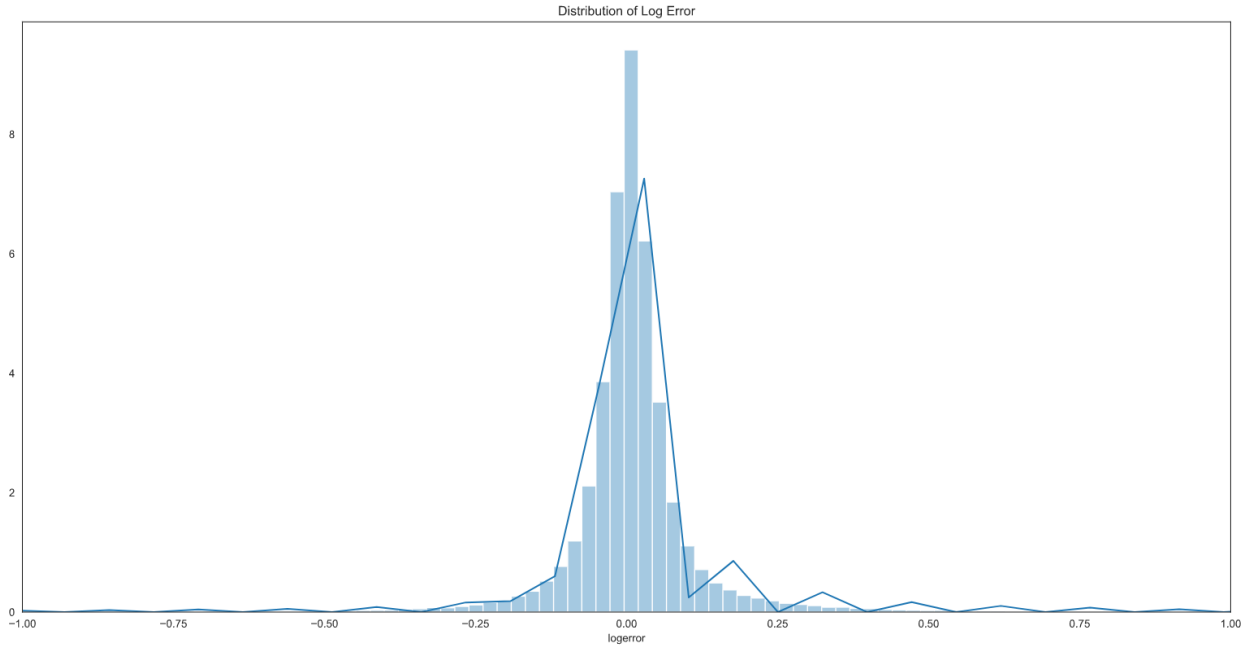
There are (3) major steps:

1. Exploratory Data Analysis (EDA)
2. Feature Engineering
3. Modeling

EDA

Log Error Distribution

First thing to do: verify if log error we are predicting is normally distributed. A lot of models are based on the assumption of normally distributed data. Verifying this will validate the models we would be using.



There is a Gaussian (normal) distribution to the log error; this means that the random sample we use to test our model will have the same distribution as our overall data, and we can guard against overfitting.

Log Error Over Time

Below is the graph showing log error over time for (3) counties:

1. Los Angeles County
2. Orange County
3. Ventura County

Ventura County records showed the highest error over time. This might indicate that features associated with location will have a big impact on our prediction.

Correlation Matrix

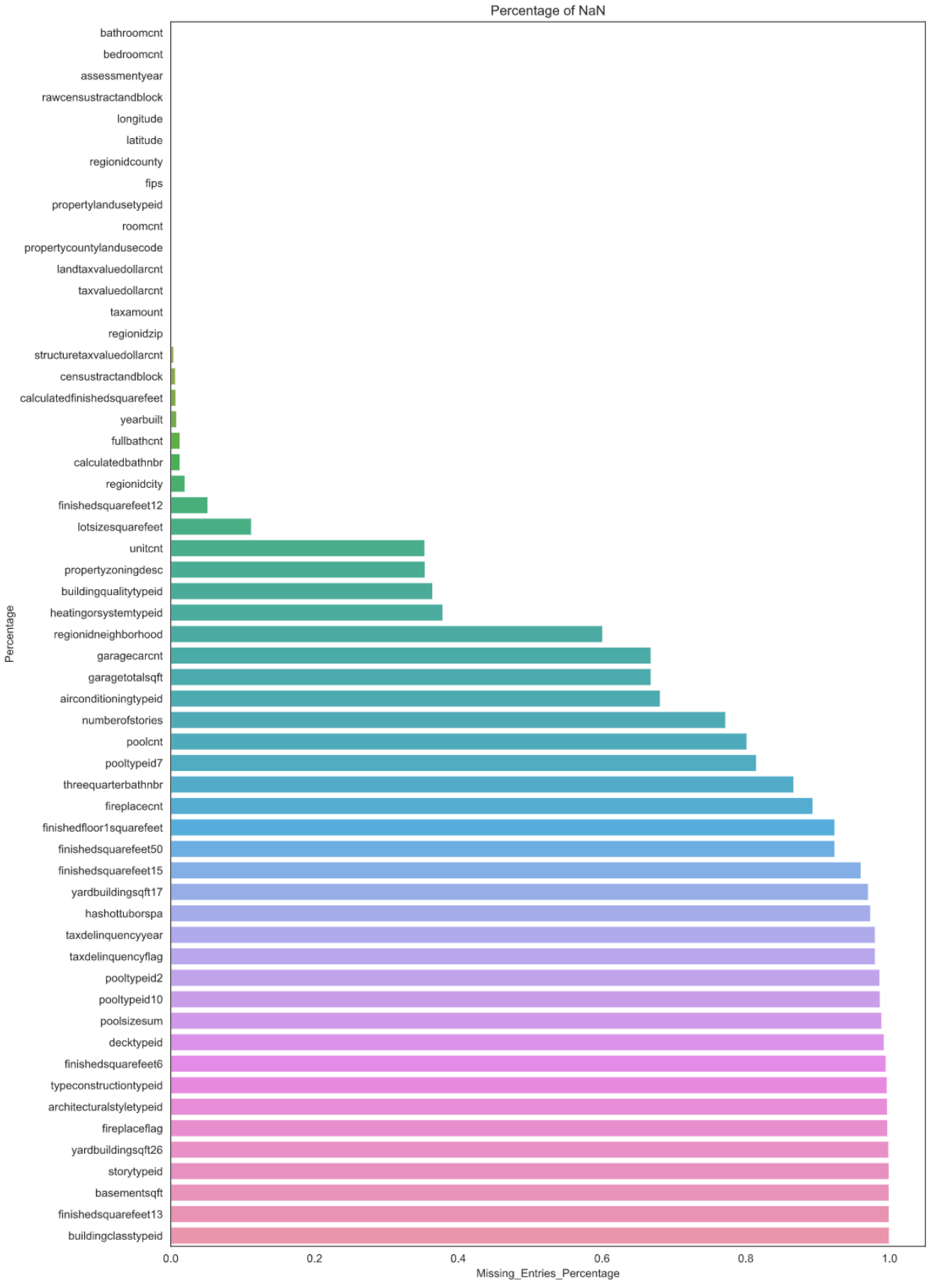
We looked at correlation of features with the absolute error and other features. We would like to see if any of the features are highly correlated.

There are no features strongly correlated with ‘logerror’. However, there are groups of features strongly correlated to each other. We would later deep dive into each group closely.

Feature Engineering

Missing Values

There are multiple features with a significant number of missing entries. Bar chart below will easily tell us how much missing entries in our data.


Percentage of NaN
bathroomcnt
bedroomcnt
assessmentyear
rawcensustractandblock
longitude
latitude
regionidcounty
fips
propertylandusetypeid
roomcnt
propertycountylandusecode
landtaxvaluedollarcnt
taxvaluedollarcnt
taxamount
regionidzip
structuretaxvaluedollarcnt
censustractandblock
calculatedfinishedsquarefeet
yearbuilt
fullbathcnt
calculatedbathnbr
regionidcity
finishedsquarefeet12
lotsizesquarefeet
unitcnt
propertyzoningdesc
buildingqualitytypeid
heatingorsystemtypeid
regionidneighborhood
garagecarcnt
garagetotalsqft
airconditioningtypeid
numberofstories
poolcnt
pooltypeid7
threequarterbathnbr
fireplacecnt
finishedfloor1squarefeet
finishedsquarefeet50
finishedsquarefeet15
yardbuildingsqft17
hashottuborspa
taxdelinquencyyear
taxdelinquencyflag
pooltypeid2
pooltypeid10
poolsizesum
decktypeid
finishedsquarefeet6
typeconstructiontypeid
architecturalstyletypeid
fireplaceflag
yardbuildingsqft26
storytypeid
basementsqft
finishedsquarefeet13
buildingclasstypeid
Percentage
0.0
0.2
0.4
0.6
0.8
1.0
Missing_Entries_Percentage

Imputation Strategy

1. Features to be grouped per category.
2. Eliminate and/or combine features containing similar information, i.e. highly correlated.
3. Impute the null values with 0 or mean value for the numerical features. Deep dive for the categorical features.

We grouped the features into:

1. Pool
2. Fireplace
3. Bathrooms
4. Tax
5. Finished square feet
6. Misc

Pool

* 'poolcnt' - Number of pools on a lot. Intuitively, 'NaN' most likely to be 0 or there's no pool for this property. 'NaN' entries will be filled with 0.

* 'hashottuborspa' - Does the home have a hottub or a spa? 'NaN' most likely to be 0 or there's no pool for this property. 'NaN' entries will be filled with 0.

* 'poolsizesum' - Total square footage of pools on property. There are (2) possible scenarios here. First, there is no pool for this property, which means the 'NaN' can be changed to 0. Second, where 'NaN' present for the properties with pool. For the second case, 'NaN' will be filled with the median value of the pool size squarefootage.

* 'pooltypeid2' & 'pooltypeid7' & 'pooltypeid10' - Type of pool or hottub present on property. To make it simpler for this purpose, these features will be dropped.

Fireplace

* 'fireplaceflag' - Does the home have a fireplace? This feature have True or False entries. 'NaN' most likely represents False. However, for this project, True and False would be better suited to be transformed to 1 and 0 respectively.
* 'fireplacecnt' - How many fireplaces in the home? 'NaN' values to be transformed to 0.

There are also cases where the 'fireplaceflag' and 'fireplacecnt' are not aligned. Below is how this case will be dealt with:

* If 'fireplaceflag' is 'True' and 'fireplacecnt' is 'NaN', 'fireplacecnt' is equal to the median value of '1'.
* If 'fireplacecnt' is 1 or larger 'fireplaceflag' is 'NaN', 'fireplaceflag' to 'True'.

Bathrooms

* 'threequarterbathnbr' - Number of 3/4 baths = shower, sink, toilet.
* 'fullbathcnt' - Number of full bathrooms - tub, sink, toilet
* 'calculatedbathnbr' - Total number of bathrooms including partials.

Again, to simplify 'threequarterbathnbr' and 'fullbathcnt' will be dropped, the assumption is that 'calculatedbathnbr' incorporates the other (2) features.

Tax

* ‘taxdelinquencyflag’, 'landtaxvaluedollarcnt', 'structuretaxvaluedollarcnt' - replace ‘NaN’ with 0.
* ‘taxdelinquencyflag’ - replace ‘Y’ with ‘1’.
* 'taxvaluedollarcnt', 'taxamount' - replace missing values with the average value for each feature.
* 'taxdelinquencyyear' - this will be dropped.

Finished SQFT

* 'calculatedfinishedsquarefeet' - eplace missing values with its average value.
* 'finishedsquarefeet13', 'finishedsquarefeet6', 'finishedsquarefeet15', 'finishedsquarefeet50', 'finishedsquarefeet6', 'finishedsquarefeet50', 'finishedfloor1squarefeet' - these are deemed to be represented by ‘calculatedfinishedsquarefeet’, thus they will be dropped.

MISCELLANEOUS

* 'storytypeid' - Numerical ID that describes all types of homes. To simplify this feature will be dropped.

* **'basementsqft' - Square footage of basement. Replace 'NaN' with '0' to represent those properties that do not have basement.**

* **'yardbuildingsqft26' - Storage shed square footage. Replace 'NaN' with '0' to represent those properties that do not have shed.**

* 'architecturalstyletypeid' - What is the architectural style of the house? To simplify this feature will be dropped.

* 'typeconstructiontypeid' - What material is the house made out of? To simplify this feature will be dropped.

* 'buildingclasstypeid' - Describes the internal structure of the home. To simplify this feature will be dropped.

* **'decktypeid' - Type of deck (if any) on property. Looks like a value is either '66.0' or 'NaN'. Replace '66.0' to 1 (to represent the properties with deck) and 'NaN' to 0 (to represent the properties with no deck).**

* 'yardbuildingsqft17' - Patio in yard. Similar to shed? this will be dropped.

* **'airconditioningtypeid' - Replace 'NaN'to '5' for 'None'.**

* 'regionidneighborhood' - Neighborhood. Could fill in blanks. Would need a key that maps lat & longitude regions with specific neighborhoods. Because 'longitude' and 'latitude' essentially provide this information, we will drop 'regionidneighborhood'.

* **'heatingorsystemtypeid' - Change 'NaN' to '13' for 'None'**

* **'buildingqualitytypeid' - Change 'NaN' to most common value.**

* **'unitcnt' - Number of units in a property. Change 'NaN' to 1.**

* 'propertyzoningdesc' - Description of the allowed land uses (zoning) for that property. Similar to 'propertylandusetypeid'. To simplify, this feature will be dropped.

* **'lotsizesquarefeet' - Area of lot in square feet. Fill 'NaN' with average.**

* 'censustractandblock' & 'rawcensustractandblock' - Census tract and block ID combined. Both features contain similar information. 'rawcensustractandblock' to be dropped.

* 'regionidcity' - City property is located in. This is redundant information, so we will drop.

* **'yearbuilt' - Year home was built. We can just fill in the 'NaN' values with the most common value.**

Modeling

Preprocessing

Outliers will be left out for our training data. In this project, the outliers are defined as log error < -0.4 or log error > 0.4.

We would still have 88,431 observations after removing the outliers.

Model Setup

We are using the PyCaret Regression module to build our model.

```
reg = setup(data = train_master,
             target = 'logerror',
             numeric_imputation = 'mean',
             categorical_features = ['propertycountylandusecode','propertyzoningdesc','fireplaceflag','taxdelinquencyflag']  ,
             ignore_features = ['parcelid','transactiondate'],
             normalize = True,
             silent = True)
```

Setup Succesfully Completed.

	Description	Value
0	session_id	8792
1	Transform Target	False
2	Transform Target Method	None
3	Original Data	(88431, 56)
4	Missing Values	True
5	Numeric Features	38
6	Categorical Features	16
7	Ordinal Features	False
8	High Cardinality Features	False
9	High Cardinality Method	None
10	Sampled Data	(88431, 56)
11	Transformed Train Set	(61901, 2095)
12	Transformed Test Set	(26530, 2095)

Model Comparison

PyCaret has a function to train all models in the model library using default hyperparameters and evaluates performance metrics with cross validation. However, in this project we are only comparing Ridge Regression, Lasso Regression, Linear Regression, LightGBM, Bayesian Ridge Regression, and Catboost.

Regression performance metrics: MAE, MSE, RMSE, R2, RMSLE, MAPE. RMSE would be the ones we are looking at.

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
0	Light Gradient Boosting Machine	0.0524	0.0066	0.0813	0.0229	0.0675	-0.1214	6
1	CatBoost Regressor	0.0524	0.0066	0.0813	0.0226	0.0677	-0.1261	20.76
2	Bayesian Ridge	0.0525	0.0067	0.0818	0.01	0.0697	-0.149	49.48
3	Ridge Regression	0.0534	0.0068	0.0822	0.0004	0.0679	-0.1005	7.034
4	Lasso Regression	0.0528	0.0068	0.0822	-0.0002	0.0707	-0.2381	2.806
5	Linear Regression	1.678e+07	2.622e+17	4.927e+08	-3.916e+19	2.181	-1.102e+08	17.2

From the comparison, LightGBM performed the best. We would use LightGBM for our predictions.

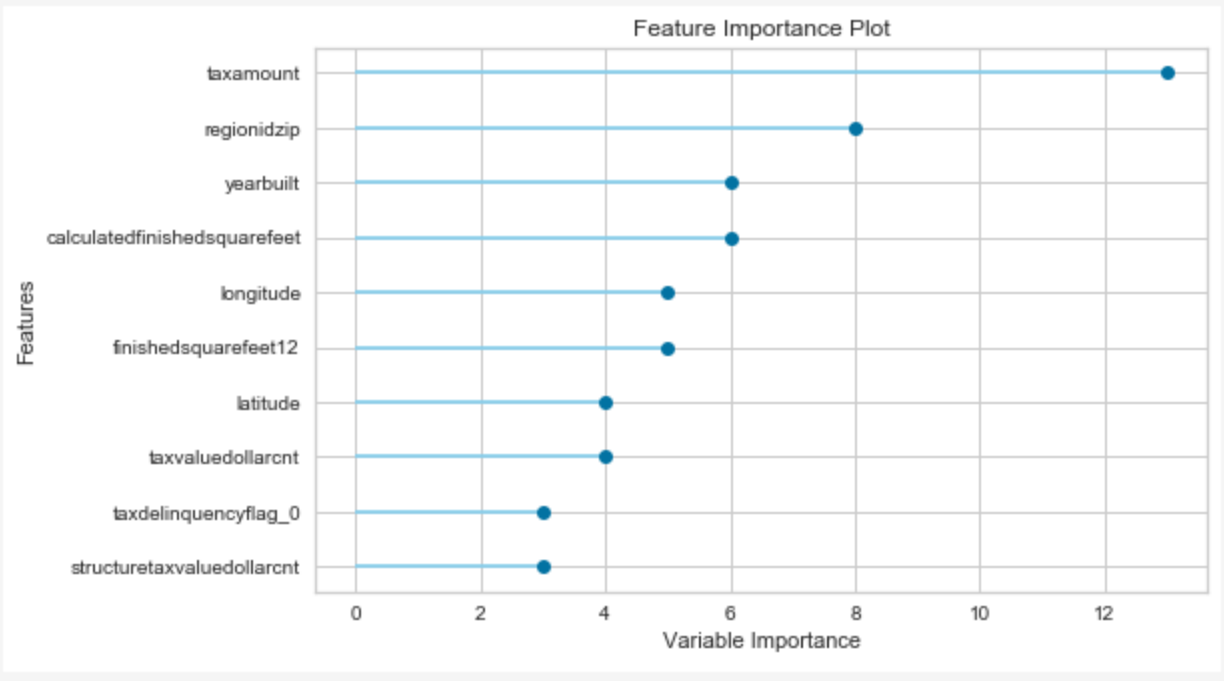
Tune Model

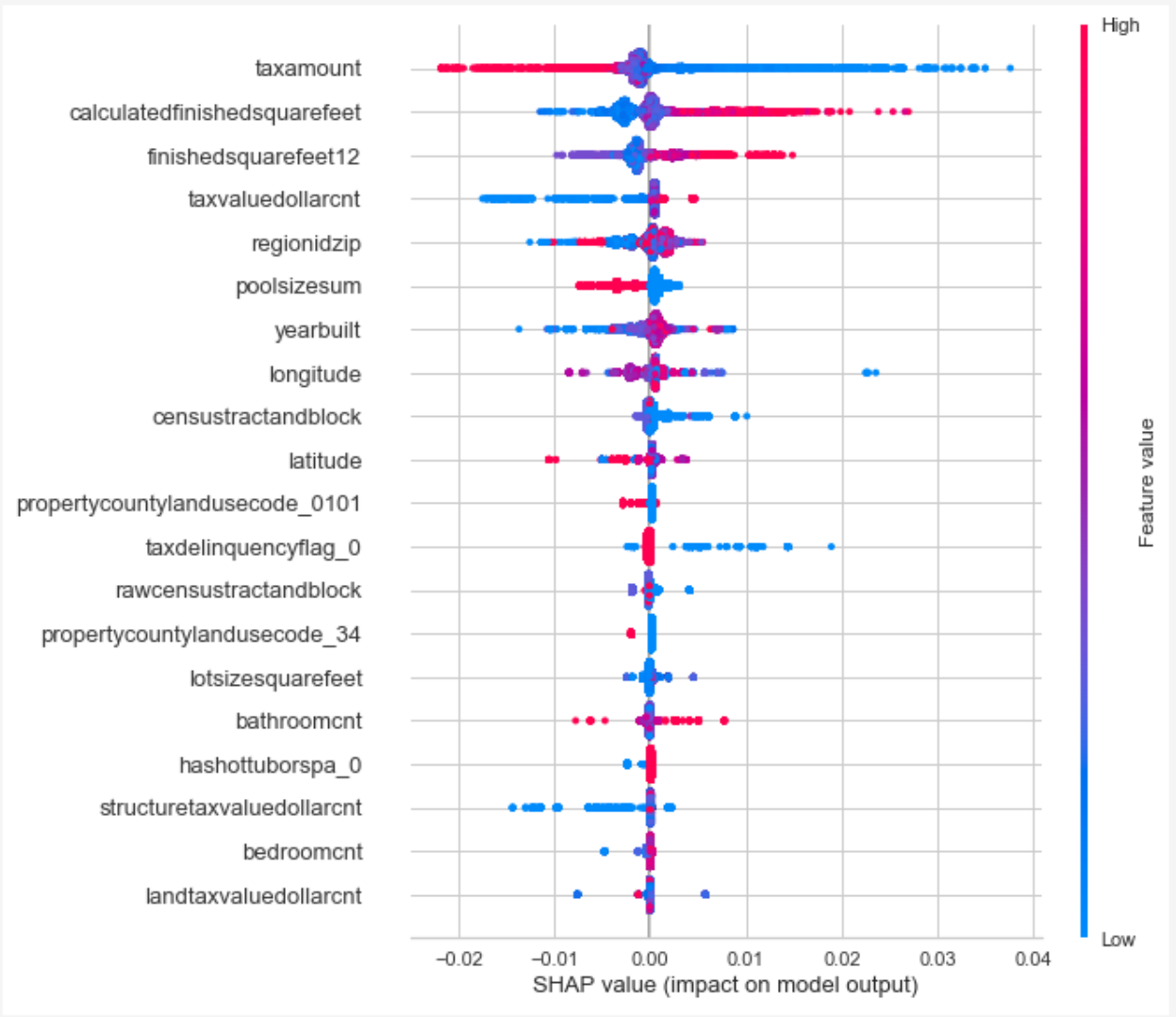
Tune the model with PyCaret function tune_model() and specify to optimize RMSE value.

```
tuned_lightgbm = tune_model(lgbm_model,  optimize='RMSE')
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	0.0521	0.0064	0.08	0.013	0.0685	-0.1825
1	0.054	0.0072	0.0851	0.0124	0.0731	-0.1234
2	0.0524	0.0066	0.0813	0.0208	0.0696	-0.1578
3	0.0526	0.0067	0.082	0.0147	0.0702	-0.2005
4	0.0524	0.0068	0.0822	0.0099	0.0702	-0.1539
5	0.052	0.0066	0.0812	0.0141	0.0692	-0.1611
6	0.0521	0.0066	0.0813	0.0127	0.0692	-0.2111
7	0.052	0.0065	0.0806	0.0155	0.0686	-0.2119
8	0.0533	0.0069	0.0832	0.0102	0.0713	-0.1249
9	0.0524	0.0067	0.0816	0.0146	0.0697	-0.1767
Mean	0.0525	0.0067	0.0818	0.0138	0.07	-0.1704
SD	0.0006	0.0002	0.0014	0.0029	0.0013	0.0305

Feature Importance





It appears from the graph that ‘taxamount’ and ‘regionidzip’ are top (2) features from our data.

- ‘taxamount’ - represent the tax influence our model the most.
- ‘regionidzip’ - represent the location is the 2nd highest feature value.

Tax and location are arguably what people look at the most when looking for a house. Those features, unfortunately, might change their influence to the price of the house.

For example, a certain area used to be expensive but due to recent natural disasters nearby the area, the house price plummeted.

Predictions

Used tuned model to predict log error.

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Light Gradient Boosting Machine	0.0527	0.0068	0.0822	0.0157	0.0701	-0.1714